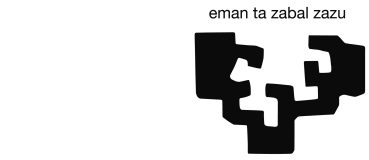


Informatika Ingeniaritzako Gradua

Konputagailuen Ingeniaritza

Gradu Amaierako Proiektua

Datu fisiologikoak eskuratzeko esperimentuak diseinatzeko sistemaren garapena: Esperimentuen diseinuaren modulua

---

Egilea

Maider Simón Aguirre

Zuzendariak

Nestor Garay-Vitoria

Edurne Larraza-Mendiluze

2016ko iraila

Laburpena

Proiektu honetan datu fisiologikoak eskuratzeko esperimentuak diseinatzeko sistemaren garapena egin da, bi modulutan banatzen dena: esperimentuak diseinatzeko modulua eta datu fisiologikoen eskuraketa modulua. Sistemaren helburu nagusia ikertzaileei subjektu esperimentalen datu fisiologikoak ingurune desberdinetan eta urrunetik jasotzeko aukera ematea da.

Esperimentuak diseinatzeko landutako moduluak hiru atal nagusi ditu. Alde batetik, web aplikazio bat garatu da ikertzaileek esperimentuak diseinatu eta kudeatu ahal izateko. Bestalde, datu fisiologikoen eskuraketa moduluarekin komunikazioa egiteko web zerbitzu (*RESTful API*) bat garatu da, eskaera mezuak *HTTP* protokoloaren bidez egiten direlarik. Azkenik, jasotako datuen biltegiraketa egiteko *HDF5* fitxategi formatua erabili da.

Aurkeztutako sistemak arlo desberdinetan izan dezake erabilera, hala nola, ikerkuntzan, medikuntzan edota kirolan.

Resumen

En el presente proyecto se ha desarrollado un sistema que permite el diseño de experimentos para recogida de datos fisiológicos, el cual se compone de dos módulos: módulo de diseño de experimentos y módulo de recogida de datos fisiológicos. El objetivo principal del sistema es ofrecer a los investigadores un modo de recoger datos fisiológicos de sujetos experimentales en diferentes entornos y de forma remota.

El módulo de diseño de experimentos implementado está compuesto por tres partes. Por un lado, se ha desarrollado una aplicación web que permite a los investigadores diseñar y gestionar experimentos. Por otro lado, para llevar a cabo la comunicación con el módulo de recogida de datos, se ha desarrollado un servicio web (*API RESTful*), cuyos mensajes de petición se realizan mediante el protocolo *HTTP*. Finalmente, para almacenar los datos recibidos se ha utilizado el formato de ficheros *HDF5*.

El sistema presentado puede tener uso en diferentes áreas, así como en la investigación, la medicina y el deporte.

Abstract

In this project a system to design experiments for gathering physiological data has been developed, which is composed by two modules: experiments design module and physiological data extracting module. The main purpose of the system is to offer researchers a way to obtain physiological data of experimental subjects in different environments and remotely.

The deployed experiments design module has three main parts. On the one hand, a web application has been developed, which allows researchers to design and manage experiments. On the other hand, to carry out the communication with the physiological data extracting module, a web service (*RESTful API*) has been developed, whose request messages are performed using the *HTTP* protocol. Finally, in order to store the received data, the *HDF5* file format has been used.

The introduced system can be used in different areas, such as in research, medicine or sport.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Aurkibidea

1.1 Motibazioa

Informatika asko eraldatu da azken urteetan, mahaigaineko ordenagailuetatik hasita nonahiko konputaziora iritsi arte. Gaur egungo teknologi berriei esker, azken urteetan seinale fisiologikoen erabilera asko zabaldu da. Sentsoreak geroz eta hedatuagoak eta txikiagoak dira, eta, ondorioz, elkarrekintza bide berriak irekitzen ari dira eta baita lor daitezkeen datuak. Elkarrekintza bide hauek janzteko gailuen (*wearable devices*) bidez lor ditzakegu.

Proiektu honetan datu fisiologikoak eskuratuz esperimentuak diseinatzeko sistema orokor bat diseinatu eta garatu nahi da, erabiltzaileen portaeran eta errendimenduan zerikusia izan dezaketen parametroak finkatu eta etorkizunean aztertu asmoz. Sistemaren helburu nagusia ikertzaileei baliabideak eta erremintak eskaintzea da, esperimentuak aurrera eramateko prozesu osoa (hauen diseinutik emaitzen analisiko fasera arte) errazteko asmoz.

Sistemaren helburua, janzteko gailuen mugikortasuna baliatuz, datu fisiologikoak ingurune erreal batean eskuratzea da. Kontuan izan behar da, jasotako datuak laborategi batean eskuratutakoak bezain zehatzak ez izan arren, ikertzaileentzat erabilgarriak izan daitezkeela, beraien naturaltasuna hobeagoa izango delakoan.

1.2 Deskribapen orokorra

Garatutako sistemak janzteko sentsore batzuen bidez pertsona baten datu fisiologikoak jaso eta zerbitzarira bidaliko ditu. Bi erabiltzaile mota identifikatu dira, alde batetik sentsoreak jantziko dituzten pertsonak (partehartzaileak edo subjektu esperimentalak) eta, bestetik, datu horiek eskuratu eta analizatuko dituzten pertsonak (ikertzaileak). Gainera, sistema ez dago mugatua erabiltzaile bakarrera, adibidez, hainbat ikertzaile egon daitezke. Zerbitzari bat egongo da sisteman partehartzaileen datuak jaso, gorde eta hauek atzitzeko eskaerak zerbitzatuko dituenak.

Sistemak bezero/zerbitzari egitura jarraitzen du, bi bezero mota desberdin daudelarik, eta ondorengo modulutan dago banatuta:

- Bezeroa:

  - Ikertzailea:
    Ikertzaileak esperimentuak diseinatzeko web aplikazio (bezero aldea) bat erabiliko du. Web aplikazio honen bidez ere, zerbitzariak biltegiratutako fitxategiak eskuratu eta aztertu ahal izango ditu.
  - Partehartzailea:
    Pertsona batek hainbat sentsore fisiologiko jantziko ditu eta lortutako datuak *Bluetooth* bidez helaraziko zaizkio bere *Android* mugikorrera. Ondoren, jasotako informazioa Internet bidez bidaliko zaio zerbitzariari, gero ikertzaileari datuak analizatzeko aukera emanez.

- Zerbitzaria:

  - Esperimentuen diseinua:
    Ikertzaileari esperimentuak diseinatzea ahalbidetuko dion web aplikazioa (zerbitzari aldea).
  - Esperimentuen emaitzen kudeaketa:
    Android aplikazioaren bidez bidalitako datuak jasotzeaz web zerbitzu bat arduratuko da. Informazioaren transmisioan segurtasuna bermatuko da protokolo seguruak erabiliz.
  - Lortutako datuen biltegiraketa:
    Jasotako datu fisiologikoen fitxategiak zerbitzarian biltegiratuko dira, *HDF5* formatuan. Dena den, etorkizuneko lan bezala, laino sistema batera pasatzeko asmoa dago.

Sistema ondorengo gailuz osatuta egongo da:

- Sentsore fisiologikoak eta hauek kudeatzeko txartela.
- Partehartzaile bakoitzeko *Android* mugikor bat sistemaren aplikazioarekin.
- Konputagailu bat, web zerbitzari bat instalatuta izango duena. Datuen transmisioaz arduratuko den web zerbitzuarekin eta ikertzaileari sarrera eskainiko dion web aplikazioarekin.
- Ikertzailearen konputagailua web aplikaziora konektatzeko eta esperimentuak kudeatzeko.

Proiektuaren irismena zabala denez, lana bi GAP-en artean banatuko da. Ezekiel Sarasua ikasleak subjektu esperimentalen elkarrekintzaren aldeko garapenaz arduratuko da, hau da, sentsore fisiologikoak eta hauek konektatzeko plakaren nondik norakoetan eta *Android* aplikazioaren garapenean egingo du lan. Maider Simón ikasleak, aldiz, zerbitzari aldeko garapenean egingo du lan: zerbitzariaren konfigurazioan, API-aren eta web aplikazioaren garapenean eta jasotako seinaleen biltegiraketan, hain zuzen ere. Bestalde, bi ikasleen artean sistema osoaren diseinua eta komunikazioa egingo da.

GAP honetan zerbitzari aldea burutzeko egin diren ikerketak eta garapena azalduko dira, proiektuaren atal komunekin batera. Beharrezkoa den ataletan ere beste GAP-eko nondik norakoak aipatuko dira.

1.3 Memoriaren egitura

Memoria honek proiektuaren nondik norakoak azaltzen ditu, hainbat kapituluetan antolatuta. 2. kapituluan Proiektuaren Helburuen Dokumentua aurki daiteke, proiektuaren plangintza eta kudeaketa azaltzen duelarik. 3. kapituluan landutako prestakuntza adierazten da eta 4. kapituluan, garapena.

Azkenik, behin proiektua garapenarekin amaitu denean, ateratako ondorioak eta etorkizunean egin daitezkeen hobekuntzak jasotzen dira 5. kapituluan. Dokumentu honen eranskinetan, garatutako sistemaren erabilpen gida eta zuzendari eta zuzendarikidearekin egindako bileren aktak bildu dira. Halaber, UCAmI'2016 [1] kongresura bidalitako eta han onartutako artikulua ere eransten da.

2. KAPITULUA

Proiektuaren Helburuen Dokumentua

Aurkibidea

Proiektuaren Helburuen Dokumentuan garatu den sistemaren deskribapena eta hau aurrera eramateko egin den plangintza azaltzen dira. Irismenaz gain, Lanaren Deskonposaketa Egitura (LDE) eta definitu diren atazak zerrendatu dira. Gainera, proiektuan zehar egin den lana islatzen duen kronograma eta dedikazio aurreikuspena aurki daitezke. Segidan, komunikazio, kalitate eta arriskuen planak eta lan metodologia zehaztu dira. Bukatzeko, benetako denbora zehazten da eta zeintzuk izan diren denboren desbiderapenaren arrazoien azterketa.

2.1 Irismena

Sistemak oso egokia den bezero/zerbitzari egitura bat jarraituko du. Esan bezala, bi bezero mota egongo dira. Entitateen arteko komunikazioa bi noranzkoetan emango da, zerbitzaria dela medio; partehartzaileak bere datu fisiologikoak (sentsoreak jantzita) zerbitzariari bidaliko dizkio, *Android* aplikazio baten bitartez. Beste aldetik, ikertzaileak subjektu esperimentalaren datu fisiologiko konkretu batzuk nahi izanez gero, esperimentua diseinatzeko fasean zehaztu beharko ditu garatuko den web aplikazioa erabiliz.

Bezeroak hainbat sentsore fisiologikoen laguntzaz bere datu fisiologikoak lor ditzake (adibidez: elektrokardiografo baten bidez bihotzaren jarduera elektrikoa). Hala ere, sentsoreen teknologiaren murriztapenak direla eta, ezinezkoa suertatzen da zuzenean zerbitzarira datuak transmititzea. Honetaz ere *Android* aplikazioa arduratuko da.

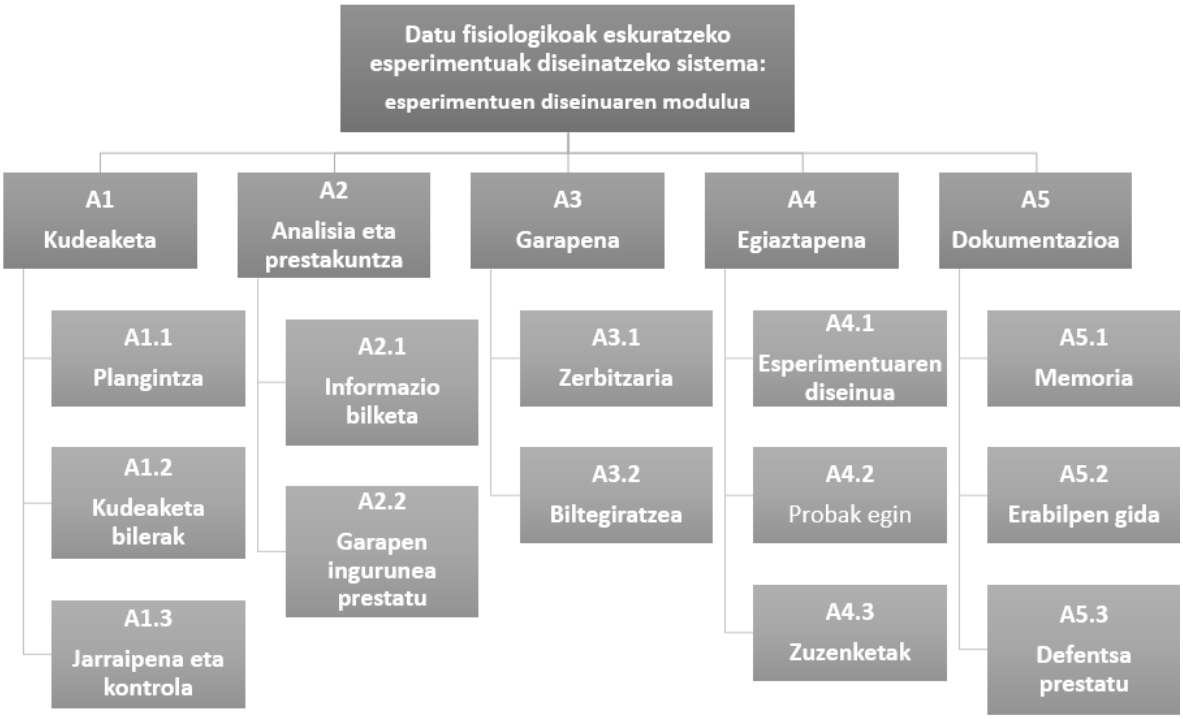
Mugikorrak sistemari funtzionalitate berriak gehitzen dizkio. Batez ere, erabiltzailearekiko elkarrekintza (jasotako datuen bistaraketa, atazen esleipena telematikoki jasotzeko aukera, etab.). Era orokor batean, esan daiteke mugikorrak hiru helburu nagusi izango dituela: sentsoreetatik datuak jaso, erabiltzaileei datuak ikuskatzeko aukera eman, eta azkenik, ikertzaileekiko komunikazioa, bai datuak bidali, bai atazak jaso.

Azkenik, zerbitzariaren atala hiru modulu nagusiez osatuta egongo da: esperimentuen diseinua, esperimentuen emaitzen kudeaketa eta lortutako datuen biltegiraketa. Lehendabiziko modulua web aplikazio baten garapenean datza eta ikertzaileekin harremanetan egongo da era zuzenean. Ikertzaileak web aplikazioaren bidez nahi duen esperimentua diseinatuko du eta partehartzaileak burutu arte itxaron beharko du. Bigarren modulua web zerbitzu baten garapenean datza eta mugikorrarekin konexioak egiteko interfazea izango da. Honen bidez, esperimentuen emaitzak zerbitzarira bidaliko dira. Azkenik, hirugarren moduluari dagokionez, laino sistema batean datu fisiologiko kantitate handiak gordetzeko egokia den formatu batean (*HDF5*) biltegiratuko dira.

Datuen iraunkortasuna eta segurtasuna kudeatzeko web zerbitzari baten inplementazioa egingo da proiektuan. Sistemaren barnean egindako datuen transmisio guztiak seguruak izango dira. Horretarako, konexio guztiak zifratu egingo dira bidalitako datuen konfidentzialtasuna bermatu ahal izateko. Pertsonen datu fisiologikoekin lan egingo denez, pertsonek dauzkaten eskubideak [2] [3] bermatuko dira.

2.2 Lanaren Deskonposaketa Egitura

2.1 irudiko diagraman proiektuan zehar landutako ataza ezberdinak ikus daitezke.



2.1 Irudia: Lanaren deskonposaketa egitura (LDE).

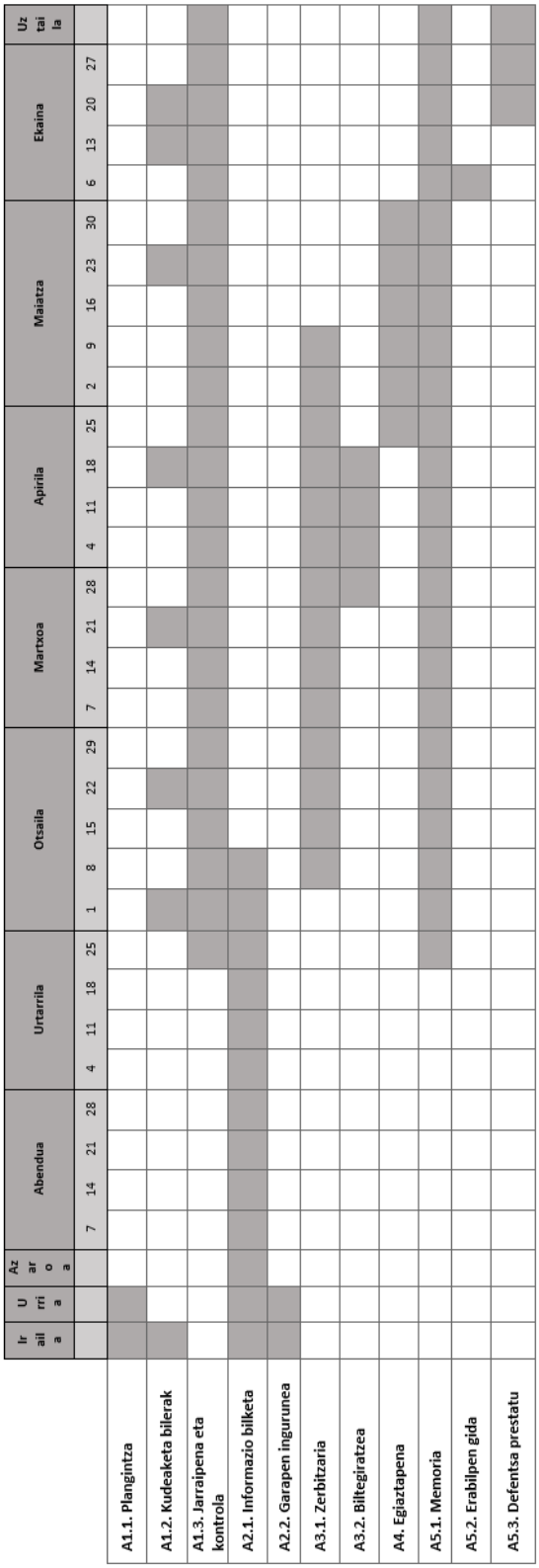
2.3 Atazak

Atal honetan proiektua aurrera eramateko beharrezkoak izan diren atazak zerrendatu dira, lanaren deskonposaketa egitura kontuan hartuz.

A1 Kudeaketa

2.4 Kronograma

Atal honetan planifikatutako lana islatzeko asmoz kronograma ikus daiteke. Kronograma hau aurreikuspen bat da eta proiektuaren garapenaren zehar beharbada aldatuko da. Kronograma asteka zatitu da, astero burutako atazak agertzen direlarik. Aipatu beharra dago proiektua kurtso osorako planifikatu dela eta, ondorioz, hasierako eta amaierako hilabeteei ez zaizkiela lan karga bera esleitu.



2.2 Irudia: Kronograma.

2.2 irudian ikusten den kronograman planifikatutako lana islatzen da. Lehenengo bi hilabeteetan konstituzio bilera eta hasierako zalantzak argitzeko beste bilera bat egin dira. Hilabete hauetan ere proiekturako plangintza landu, informazioa bilatzen hasi eta garapen ingurunea prestatu dira.

Otsaileko erdialdera arte informazioa bilatzen igaroko da denbora gehiena. Behin alde teorikoa ondo barneratuta izanda, zerbitzari aldearen garapena egingo da. Honekin batera memoria idazten hasi eta egindako lanaren jarraipen eta kontrola ere landuko da proiektua amaitu bitartean. Otsaileko azken astetik aurrera, behin zerbitzari aldea erabilgarri dagoenean, proiektukideak garatutako Android aplikazioaren hasierako bertsioarekin egingo da lan zerbitzariarekin konexioa ezartzeko. Martxoaren amaiera aldera proiektuaren biltegiratze moduluan egingo da lan, hau da, datu fisiologikoak fitxategietan gordetzeko formatuak aztertu eta hauek sortzeko eta kudeatzeko web aplikazioa moldatuko da.

Behin banakako garapena bukatuta dagoela, apirila eta maiatzean, proiektukidearekin batera lan egingo da sistemaren balidazio fasean. Behin sistema osoa amaituta dagoenean, honen erabilpen gida idatziko da, material gehigarri eta lagungarri gisa. Azkenik, ekaina amaieran eta uztaila hasieran defentsarako materiala prestatuko da. Esan beharra dago proiektuaren garapenaren zehar hainbat kudeaketa bilera egingo direla, bai zalantzak argitzeko nola zuzendariak eta zuzendarikideak proiektuaren egoeraren berri jakin dezaten.

2.5 Dedikazio aurreikuspena

2.1 taulan proiektua garatzeko egin den dedikazio aurreikuspena ikus daiteke, definitutako lan-paketeen arabera.

2.1 taulan ikus daitekeen moduan, A3 lan-paketea, hau da, garapenari dagokiona, izango da dedikazio gehien behar duena. Hurrengoak, A5 (dokumentazioa) eta A4 (egiaztapena) lan-paketeak izango dira, A2 (analisia eta prestakuntza) lan-paketearekin jarraituz. Azkenik, A1 (kudeaketa) lan-paketea denbora gutxien behar izango duela espero da.

2.6 Komunikazio plana

Zuzendari eta zuzendarikidearekin bi modutara egingo da komunikazioa. Alde batetik, zalantza edo arazoren bat izanez gero, EHU-ko posta elektronikoa erabiliko da hauek

Lan-paketea	Dedikazio aurreikuspena (ordutan)
A.1. Kudeaketa	20
A1.1. Plangintza	10
A1.2. Kudeaketa bilerak	5
A1.3. Jarraipena eta kontrola	5
A2. Analisia eta prestakuntza	45
A2.1. Informazio bilketa	40
A2.2. Garapen ingurunea prestatu	5
A3. Garapena	120
A3.1. Zerbitzaria	110 *
- A3.1.1. Zerbitzaria konfiguratu	10
- A3.1.2. Web zerbitzua	40
- A3.1.3. Web aplikazioa	60
A3.2. Biltegiratzea	10
A4. Egiaztapena	70
A4.1. Esperimentuaren diseinua	20
A4.2 Probak egin	20
A4.3 Zuzenketak	30
A5. Dokumentazioa	103
A5.1. Memoria	90
A5.2 Erabilpen gida	3
A5.3. Defentsa prestatu	10
GUZTIRA	**358**

* Zerbitzariko hiru azpiatazen batura metatua.

2.1 Taula: Aurreikusitako dedikazioa.

argitzeko. Bestetik, proiektuaren garapen-egoeraren arabera hainbat bilera egingo dira Informatika Fakultatean. Proiektuarekin hasi aurretik lehenengo bilera bat proiektuaren atal garrantzitsuenak definitzeko eta finkatzeko eta hasierako zalantzak argitzeko. Hurrengo bilerak behar direnean egingo dira, proiektuan aurreratu ahala. Azkenik, itxiera bilera bat egingo da proiektua bukatutzat eman ahal izateko.

Bestalde, gure artean hainbat modutara komunikatuko gara. Bereziki, *WhatsApp* bidez hitz egingo dugu zalantzak azkar argitu eta bakoitzak egiten ari den lanaren berri izateko. EHU-ko posta elektronikoa ere erabiliko dugu kasu batzuetan.

Esan beharra dago proiektuaren inguruan sortzen zaizkigun zalantzak argitzeko Borja Gamecho doktorearen [4] laguntza ere edukiko dugula.

2.7 Kalitate plana

Atal honetan proiektuaren kalitate plana azalduko da, proiektuaren kalitate betekizun eta dimentsioak definituz. Honakoak dira proiektuaren oinarrizko betekizunak:

- Sentsoreetatik eskuratutako datuak ahalik eta zehatzenak izatea (elektromiograma, elektrokardiograma, azalaren jarduera elektrikoa, argitasuna, azelerometroa, ...).
- Erabiltzaileak sentsoreen egokitasuna egiaztatzeko aukera.
- Ikertzaileak jasotako datuak sentsoreetatik lortutakoekin bat etortzea (datuen integritatea).
- Erabiltzaile eta ikertzaileen kontuen kudeaketa zuzena.
- Ikertzaileari datuak era ordenatu eta ulergarri batean helaraztea.
- Ikertzaileek beharrezkoak dituzten datuak zeintzuk diren erabiltzaileei adieraztea eta ebaluatzeko mekanismo bat izatea (atazen esleipena).
- Lortutako datu fisiologikoen iraunkortasuna datu-base batean.
- Erabiltzaile kontuak eta datu fisiologikoak biltegiratzerakoan Datuen Babeserako Lege Organikoa eta Etika Batzordearen eskakizunak bete.

Ondorengoak dira proiektuaren kalitate dimentsioak:

- Sistema, datu fisiologikoak eskuratzen dituen txartel zehatz batera mugatuta ez egotea.
- Ikertzaileak, erabiltzaileak sentsoreetatik jasotako datu fisiologikoak denbora errealean ikusteko aukera.
- Erabiltzaile eta ikertzailearen interfazeak erabilgarriak eta irisgarriak izatea.
- Erabiltzaileek egindako atazak eta ebaluazioen atazak berrikusteko aukerak.

2.8 Arriskuen plana

Jarraian proiektuaren garapenean zehar gerta litezkeen arazoak aurreikusi eta hauek nola konpondu planifikatu da. Esan beharra dago arrisku guztiek ez dutela modu berdinean eragina izango proiektuarengan.

1. Ustekabe larriak ekiditeko proiektua entregatze-epea baino 15-30 egun lehenago bukatuko da, hala ere, ustekaberen bat izanez gero, galdutako orduak hurrengo asteetan berreskuratuko dira era uniforme batean. Proiektua bi pertsonen artean banatuta dagoenez muturreko ustekabe batean beste pertsona egin daiteke kargu. Arazo epe luzekoa izanez gero planifikazioa berregingo da, atazen lehentasun eta denborak birkalkulatuz.

2. Horren ildotik, denbora falta kudeatzeko asmoz proiektuaren amaiera benetako epea baino 15-30 egun lehenago planifikatuko da. Horrela, egun batzuk utziko dira koltxoi gisa. Arazoa gertatu eta konponduz gero, kalitate dimentsioak albo batera utziko dira oinarrizko betekizunetan zentratzeko.

3. Analisia eta prestakuntza atalean denbora nahikoa emango da garapenean zehar gerta daitezkeen arriskuak ekiditeko. Hala ere, arazoren bat izanez gero zuzendariarengana, irakasleengana edota ikertzaileengana joko da.

4. Datu galerak ekiditeko egunero babes kopiak egingo dira bai lan egiteko erabiltzen den makinan bertan nola kanpoko euskarri batean.

5. Hardwarean matxuraren bat izanez gero, proiektua unibertsitatean dauden baliabideekin moldatuko da.

2.9 Eskuraketen kudeaketa

Proiektu honen garapena hasi aurretik eskuraketa bat egin beharko da, bereziki, hau da, sentsoreak eta hauek konektatzeko plaka lortzea. Eskuraketa hau Borja Gamechoren bidez egingo da.

Gainerako beharrezko eskuraketak Internet bidez egingo dira, hala nola erabiliko den software-a eta erreminta desberdinak eta hauen dokumentazio orriak.

2.10 Lan metodologia

2.1 atalean azaldu den moduan, proiektu hau hiru ataletan banatu da, bi banakako atal eta atal amankomun bat. Laburbilduz:

1. Ezekiel Sarasua ikasleak: txartela, sentsoreak eta *Android* aplikazioaren garapena.

2. Maider Simón ikasleak: zerbitzariaren konfigurazioa, web zerbitzua eta web aplikazioaren garapena eta datu fisiologikoen biltegiraketa.

3. Atal amankomuna: sistemaren arkitekturaren eta moduluen arteko komunikazio protokoloen diseinua.

Ezekielek partehartzaileen atalan egingo du lan, hau da, erabiliko den plakaren funtzionamendua eta sentsoreak aztertu eta *Android* aplikazioa garatuko du. Maider, berriz, zerbitzari atalan zentratuko da, hau da, web zerbitzariaren instalazioan, konfigurazioan eta web aplikazioaren inplementazioan egingo du lan. Bi atal hauen arteko elkarrekintza bien artean adostuko da, hau da, sistemaren egitura nagusiaren diseinua, arkitektura eta bi atalen komunikazio protokoloen diseinua.

Nahiz eta bakoitza bere zatian zentratu, biek izango dute bestearen lanaren berri momentu guztian. Horretarako, Informatika Fakultateko 302. laborategian egingo dira bilerak, egindako lana batak besteari erakutsi eta azaltzeko. Bestalde, bakoitzak bere kasa egingo du lan eta proiektuan aurrera egiteko arazoak edo zalantzak sortuko balira, proiektuko zuzendari eta zuzendarikideari galdetuko liekete.

Proiektuaren garapena hainbat fasetan banatu da. Lehenengoan, Proiektuaren Helburuen Dokumentua sortuko da plangintza moduan, hau da, zer egin behar den eta nola dokumentatu. Bigarren fasean, lana aurrera eramateko behar den informazioa bilatu, ikasi eta dokumentatuko da. Hirugarren fasean, ikasitakoa aplikatuko da sistemaren inplementazioa egiteko. Azkenik, garatutako sistema probatuko du hainbat proba kasu aztertzen.

Esan beharra dago fase bakoitzean egindako guztia dokumentatuko dela, gero GAP memorian txertatu eta moldatzeko.

2.11 Jarraipena eta kontrola

Atal honetan planifikatutakoa benetako denborarekin alderatuko da, izandako desbiderapen nagusiak aipatuz eta hauen zergatiak azalduz.

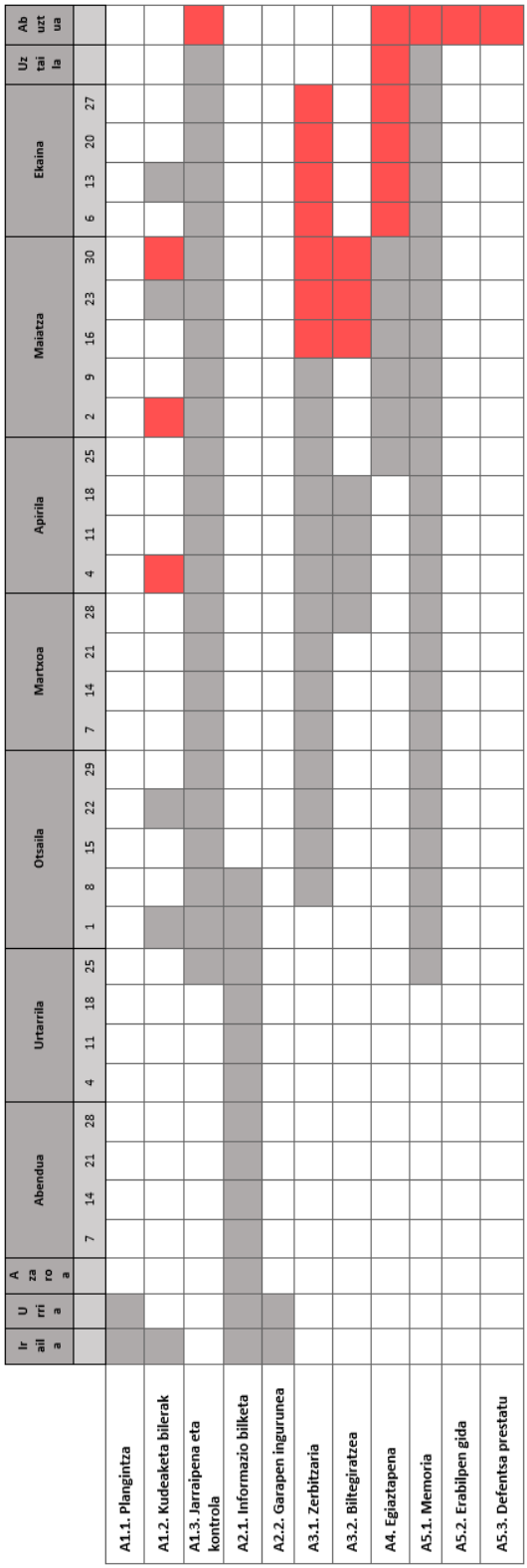
2.11.1 Benetako denbora

2.5 atalean proiektuari dedikatzea aurreikusi zen denbora azaldu da. Orain, behin proiektua amaituta benetako denbora zein izan den aztertuko da 2.2 taularen eta 2.3 kronogramaren bidez.

Lan-paketea	Dedikazio aurreikuspena (ordutan)	Benetako dedikazioa (ordutan)
A.1. Kudeaketa	20	28
A1.1. Plangintza	10	10
A1.2. Kudeaketa bilerak	5	15
A1.3. Jarraipena eta kontrola	5	3
A2. Analisia eta prestakuntza	45	55
A2.1. Informazio bilketa	40	50
A2.2. Garapen ingurunea prestatu	5	5
A3. Garapena	120	140
A3.1. Zerbitzaria	110 *	120 *
- A3.1.1. Zerbitzaria konfiguratu	10	10
- A3.1.2. Web zerbitzua	40	45
- A3.1.3. Web aplikazioa	60	65
A3.2. Biltegiratzea	10	20
A4. Egiaztapena	70	90
A4.1. Esperimentuaren diseinua	20	20
A4.2 Probak egin	20	30
A4.3 Zuzenketak	30	40
A5. Dokumentazioa	103	103
A5.1. Memoria	90	90
A5.2 Erabilpen gida	3	3
A5.3. Defentsa prestatu	10	10
GUZTIRA	**358**	**416**

* Zerbitzariko hiru azpiatazen batura metatua.

2.2 Taula: Benetako denbora.



2.3 Irudia: Benetako kronograma.

2.11.2 Desbiderapenen azterketa

Atal honetan desbiderapenen azterketa egingo da, gertatu diren desbiderapenen zergatiak azalduz eta aurreikusitako denbora eta benetako denbora alderatuz.

2.5 taulari erreparatuz, proiektua amaitzeko guztira 58 ordu gehiago behar izan dira aurreikusitako denborarekin alderatuz. Honen arrazoiak hainbat izan dira. Alde batetik, garapenarekin hasi aurretik egin beharreko analisia eta prestakuntza atazak aurreikusitako denbora baino 10 ordu gehiago behar izan du. Honen arrazoia erabilitako teknologien eta erreminten buruzko ezjakintasuna izan da, batez ere. Beste aldetik, proiektuko garapenean zailtasunak egon dira, batez ere zerbitzariaren inplementazioan. Izan ere, web garapenean izandako esperientzia gutxi zenez, denbora gehiago behar izan da kodean zuzenketak egin ahal izateko.

Bestalde, 2.3 kronograman denboran zehar edukitako desbiderapenak gorriz markatu dira. Lehen aipatutako zailtasunak direla eta, proiekturako definitutako ataza gehienak atzerapen bat jasan dute. Zailtasun hauez gain, arazo pertsonalak direla eta proiektuaren garapena atzeratu behar izan da. Kronograma honetan izandako zailtasun guzti hauek islatu dira.

Zerbitzariaren garapena maiatzaren lehenengo astean amaitu ordez, ekainara arte luzatu da. Biltegiratze atazan ere arazoak eduki dira. Inplementatu zen programaren hasierako bertsioaren portaera ez zen egokia sistema osoaren egiaztapen prozesua egin zenean eta, horregatik, maiatzan zehar zuzendu da. Ildo berean, UCAmI [1] kongresurako artikulua idazteko sistema osoaren egiaztapena egin behar izan da ekainan. Ondorioz, egiaztapen ataza maiatzan zehar prestatu eta ekainara arte luzatu da eta kudeaketa bilera gehiago egin behar izan dira. Atzerapen guzti hauek direla eta, erabilpen gidaren idazketa, memoriaren azkeneko berrikuspenak eta defentsaren prestakuntza abuztura pasa behar izan da.

3. KAPITULUA

Proiektuaren prestakuntza

Proiektu honetan erabili beharreko teknologiak sakonki ezagutzeko behar diren bilaketak eta azterketak adierazten dira atal honetan.

Aurkibidea

3.1 Web zerbitzuak

Atal honetan web zerbitzuen inguruan egin den azterketa azaltzen da. Lehenik eta behin, web zerbitzuak zer diren eta hauen ezaugarri orokorrak aipatzen dira. Bigarrenik, *SOAP* eta *REST* motako web zerbitzuei buruz sakondu da, bakoitzaren ezaugarriak eta aukerak aztertuz. Azkenik, atala amaitzeko, web zerbitzu hauek inplementatzeko aukera desberdinak komentatzen dira, laburki.

3.1.1 Sarrera

Web zerbitzu (*Web Service*) bat aplikazio batek eskaintzen duen zerbitzu bat da, non bere logika edozein plataformako bezeroei atzigarri uzten dien, interfaze baten bidez eta Interneteko protokolo estandarrak erabiliz [5] [6] [7]. Web zerbitzuek aplikazioen arteko komunikazioa ahalbidetzen dute, garatzeko lengoaia edo plataforma kontuan hartu gabe.

Web zerbitzuak web zerbitzari baten bidez atzitzen dira, plataformarekiko modu independentean eta protokolo estandarrak erabiliz (*HTTP*, *SOAP*, *WSDL*, *UDDI*, *REST*, etab.).

Web zerbitzuak oso erabilgarriak izan daitezke kasu batzuetarako. Adibidez, garatutako aplikazio bat, bere datu eta prozesu guztiekin, beste plataforma batzuetara zabaltzeko asmoa badago, interesgarria izan daiteke web zerbitzu bat inplementatzea. Honi esker, egindako lan guztia ez genuke galduko. Izan ere, web zerbitzuak tarteko geruza baten antzera lan egiten du, eta gure aplikazioa plataforma berriekin komunikatzeko bidea errazten du.

Web zerbitzuen inplementazioan aukera asko daude gaur egun, hainbat protokolo, formatu, lengoaia eta *framework* desberdinak erabil daitezkelako. Normalean *PHP* lengoaia erabiltzen da web zerbitzuak programatzeko eta hauen garapena errazteko *framework*-ak daude, funtzio eta metodo erabilgarriak dituzten interfazeak. Datuen transmisiorako formatu erabilienak *XML* eta *JSON* dira, gaur egun batez ere azkeneko hau erabiltzen delarik. Izan ere, *JSON* formatuak abantaila asko ditu *XML*-rekin alderatuz. Aukera hauetan sakonduko da memoria honen hurrengo ataletan.

Ezaugarri orokorrak

Web zerbitzuek elkarreragiletasuna eskaintzen dute software aplikazioen artean, hauen ezaugarriak zeintzuk diren edo zer plataformetan instalatuta dauden kontuan hartu ga-

be. Testuan oinarritutako estandarrak eta protokoloak sustatzen dituzte, eta, hauei esker, edukia modu errazago batean atzi eta bere funtzionamendua uler daiteke. Gainera, toki desberdinetan kokatuta dauden enpresa desberdineko software eta zerbitzuak erraz konbinatzea ahalbidetzen dute, zerbitzu integratuak eskaini ahal izateko.

Dena den, arazoak ere badaude. Errendimendua baxua da konputazio banatuarekin alderatuta, testuan oinarritutako formatua erabiltzeagatik. Izan ere, *XML* erabiltzen bada, metadatu asko bidaltzen dira.

Lehen esan moduan, web zerbitzu bat inplementatzerako orduan, hainbat aukera daude. Memoria honetan bi aipatzen dira, nagusienak eta gehien erabiltzen direnak, *SOAP* eta *REST*, alegia. 3.1.2 eta 3.1.3 ataletan hauetan sakonduko da.

3.1.2 *SOAP*

SOAP (***S**imple **O**bject **A**ccess **P**rotocol*) protokoloa web zerbitzariak datuak jaso eta bidali ahal izateko erabiltzen da [5]. Web zerbitzuak inplementatzeko modu tradizionala dela esan daiteke. Izan ere, *W3C* (*World Wide Web Consortium*) [8] estandarra da, *XML*-n (***EX**tensible **M**arkup **L**anguage*) [9] oinarrituta dagoena. *SOAP* zerbitzu batek itzulitako emaitzak (erantzunak) *XML*-n oinarrituta daude.

Ezaugarriak

SOAP web zerbitzu oso sendoak inplementa daitezke, modu errazan. Gainera, lengoaia askotarako euskarria dute, beste batzuen artean, *.NET* [10], *PHP* [11] eta *Java* [12]. *SOAP* oso erabilia eta zabalduta dagoenez, dokumentazio asko aurki daiteke. *SOAP* protokoloa estandarra da eta, honi esker, erabiltzeko arauak ondo definituta daude.

Dena den, arazoak ere baditu. Izan ere, informazio bidalketan *XML* erabiltzen da eta, ondorioz, metadatu asko bidali behar izaten dira. Mugikorretarako zerbitzuen kasuan, banda zabalera asko kontsumitzen da. Horregatik, *SOAP* zerbitzuak aproposak dira zerbitzari berean (domeinua partekatzen dutenak) lan egiten duten web zerbitzuen kasuan, banda zabaleraren erabilera eta atzerapena (latentzia) ahalik eta gehien murriztearren.

Egitura

Mota honetako web zerbitzuetan ondorengo protokolo estandarrak erabiltzen dira: *HTTP* (***H**yper**T**ext **T**ransfer **P**rotocol*) [13], *SOAP*, *WSDL* (***W**eb **S**ervices **D**escription **L**anguage*) [14] eta *UDDI* (***U**niversal **D**escription, **D**iscovery and **I**ntegration*) [15], beste batzuen artean. Web zerbitzu hauek osatzen duten geruza-multzoa ondorengoa da:

***UDDI* [Aurkitu]**
Interneteko zerbitzuen direktorio bat eskaintzen du. Internet bidez aplikazioen arteko komunikazioa ahalbidetzen du.

***WSDL* [Definitu]**
Zerbitzuak definitzeko modu bat eskaintzen du. *XML*-n oinarrituta dago eta web zerbitzua atzitzeko moduak definitzen ditu. hau da, web zerbitzuak eskaintzen dituen interfazeak eta beharrezko datu motak zeintzuk diren erakusten du.

***SOAP* [Deitu]**
Zerbitzuen metodoak deitzea ahalbidetzen du. Protokolo honen egitura objektu hauetaz dago osaturik:

- ***Container* (edukia)**: web zerbitzuaren informazio guztia gordetzen du.
  - *Header* (goiburukoa): web zerbitzuaren informazioa gordetzen du.
  - *Body* (gorputza): web zerbitzua osatzen duten elementuen deskribapena gordetzen du.

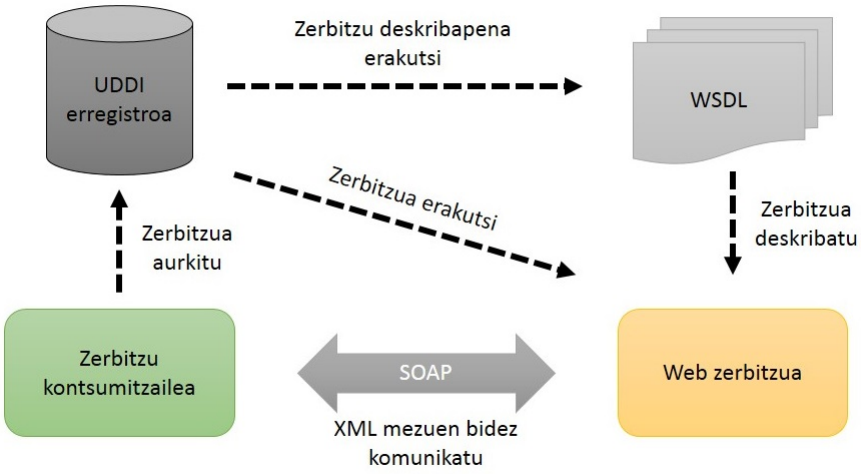
***XML* [Datuak]**
Zerbitzuen kontsumitzaileei mezuak bidali eta jasotzea ahalbidetzen du.

***HTTP*, *SMTP* (*Simple Mail Transfer Protocol*) [16], *TCP* (*Transmission Control Protocol*) [17], etab. [Garraioa]**
Goiko geruzei euskarria ematen die.

Funtzionamendua

Behin azalduta zer protokolok hartzen duten parte *SOAP* motako web zerbitzuetan, protokolo hauek elkarren artean nola funtzionatzen duten ikusiko da. 3.1 irudian ([18] erreferentzian agertzen den irudian oinarrituta) elkarrekintza hau azaltzen da.

1. Web zerbitzuak *WSDL*-an uzten du zerbitzuaren deskribapena eta *UDDI* direktorioan erregistratzen du.
2. Zerbitzu kontsumitzaileak (bezeroa) web zerbitzua eskatzen du eta *UDDI* erregistroa kontsultatzen du web zerbitzua aurkitzeko.
3. Bezeroak, *WSDL*-aren deskribapena begiratuz, eskaera-mezu bat bidaltzen dio zerbitzariari *SOAP* protokoloaren bidez (*XML* mezuak).
4. Web zerbitzuak *SOAP* eskaera-mezua analizatzen eta prozesatzen du eta, honen arabera, ekintza bat exekutatzen du. Emaitza *SOAP* formatuan idazten da eta bezeroari bidaltzen zaio.
5. Bezeroak mezua analizatzen eta interpretatzen du, nahiz eta erroreren bat gertatu.



3.1 Irudia: *SOAP* protokoloen arteko elkarrekintza.

3.1.3 *REST*

REST (***RE****presentational* ***S****tate* ***T****ransfer*) gaur egun oso zabalduta dagoen protokoloa eta arkitektura estilo bat da [19]. Lan egiteko modua askoz sinpleagoa eta malguagoa da *SOAP* protokoloarekin alderatuz, ez baitauzka hainbeste arau. Datuak zuzenean tratatzen direnez, askoz metadatu gutxiago bidaltzen dira.

RESTful web zerbitzu bat, beraz, *REST* protokoloa jarraituz sortzen den aplikazioa da. Termino desberdinei dagokienez, *REST* eta *RESTful* direlakoek gauza bera adierazten dute. Desberdintasun bakarra *REST* terminoa protokoloari erreferentzia egiten diola

da eta *RESTful*, aldiz, *REST* protokoloa jarraitzen duen web zerbitzu bati egiten diola erreferentzia. *HTTP* protokoloa erabiltzen da sistema arteko komunikazioa ezartzeko eta baliabideak maneiatzeko. *REST* web interfaze estandarra eta sinplea da.

Enpresa askok *REST* bidez definitzen dituzte haien aplikazioen *API*-ak (***A**pplication **P**rogramming **I**nterface*) [20]. *REST* erabiltzea gomendagarria da kanpoko aplikazioetarako, hau da, domeinu desberdin batean dagoen webgunea edota gailu mugikorrekin lan egiten ari bagara (*RESTful* zerbitzu batetik datozen datuak kontsumituko dira).

Ezaugarriak

REST arkitektura estilo bat da eta baliabideekin egiten da lan. Baliabide bat jaso nahi den informazioa adierazten duten datuek osatzen dute. *REST*-ek aplikazio arteko baliabide-transferentzia estandarizatzen du. Adibidez, datu-base baten erregistro bat edota sistemako erabiltzaile bakoitza baliabideak izango lirateke. Ildo berean, baliabide multzo bati bilduma deritzo. Beraz, erabiltzaile-multzo bat jasotzean erabiltzaile bilduma bati buruz hitz egiten egongo ginateke.

REST SOAP protokoloa baino aukera sinpleagoa da. Datuen bidalketan hainbat lengoaia edo formatu desberdin erabil daitezke: *XML*, *JSON* [21] edota testu arrunta. Malgutasun honi esker, oso metadatu gutxi bidali behar dira, batzuetan ia ezer. Banda zabalera askoz hobeto aprobetxatzen da eta latentzia murrizten da. Programatzerako orduan ere erraztasun hauek nabaritzen dira eta, ondorioz, *RESTful* web zerbitzu baten garapenaren konplexutasuna eta denbora murrizten dute.

REST egoerarik gabeko bezero/zerbitzari protokolo bat da. *HTTP* mezuek eskaera bakoitza ulertzeko informazio guztia daukate. Ondorioz, ez bezeroak ez zerbitzariak komunikazioan zehar ez dute egoerarik gorde behar. Dena den, praktikan, aplikazio askok saioaren egoera gordetzeko *cookie*-ak edo beste motako tresnak erabiltzen dituzte. Kontuan hartu behar da *REST*-ek ez duela *URL*-en (***U**niversal* **R**esource **L**ocator) [22] berridazketa onartzen.

Egitura

HTTP metodoak

REST-en, *HTTP* metodoak erabiltzen dira erregistro edo baliabide baten gainean eragiketak egin ahal izateko. Metodo nagusienak *CRUD* [23] eragiketak bezala errepresentatzen dira, hau da, *Create* (sortu), *Read* (irakurri), *Update* (eguneratu) eta *Delete* (ezabatu). *HTTP* metodo garrantzitsuenak *POST*, *GET*, *PUT* eta *DELETE* dira, baina *PATCH* eta *HEAD* metodoak ere badaude [24].

Metodo bakoitzean sakonduko dugu, funtzioak eta erabilera azalduz, adibideekin batera:

GET

Baliabide edo bilduma baten eskaera egiteko *HTTP* metodoa da, defektuz erabiltzen dena web eskaeretan. Egindako eskaerak irakurri besterik ezin dira egin, aldaketarik egin gabe. Nahiz eta eskaera hainbat aldiz errepikatu, beti emaitza bera jaso beharko genuke. Esaterako, sistemako erabiltzaile guztiak ikusteko, honako eskaera egingo genuke:

```
GET /api/erabiltzaileak
```

POST

POST metodoa baliabide berri bat sortzeko erabiltzen da. Adibidez, erabiltzaile berri bat sortzeko eskaera URI honetara egin beharko genuke. Eguneratuko den informazioa formatu desberdinetan bidal daiteke, *JSON*, esaterako.

```
POST /api/erabiltzaileak
```

PUT

PUT metodoa baliabide bat eguneratu nahi denean erabiltzen da. Baliabidea URI zehatz baten bidez identifikatzen da, normalean baliabidearen izena eta identifikadore kode baten bidez:

```
PUT /api/erabiltzaileak/535
```

DELETE

DELETE metodoa baliabidea guztiz ezabatu nahi denean erabiltzen da. URI-aren bitartez

```
DELETE /api/erabiltzaileak/535
```

PATCH

PATCH metodoak baliabide bat partzialki eguneratzen du, hau da, baliabidearen atributu batzuk bakarrik eguneratzen ditu. PUT-en antzekoa da, baina atributu bakarra eguneratu nahi denean, adibidez, askoz eraginkorragoa da PATCH erabiltzea, banda zabalera gutxiago behar baitu.

```
PATCH /api/erabiltzaileak/535
```

HEAD

HEAD metodoa GET metodoaren antzekoa da, baina eskaeraren emaitzan *HTTP* goiburukoak eta *HTTP* kodea bueltatzen ditu. Metodo hau normalean nabigatzaileak erabiltzen du dokumentu bat existitzen den edo aldatu den ala ez egiaztatzeko. Loturak egiaztatzeko erabilgarria izan daiteke, GET eskaerak egin beharrean, azkarragoa baita.

HTTP metodoen ekintzak laburbildu dira 3.1 taulan:

Metodoa	Ekintza
GET	Baliabide edo bilduma bat eskuratu
POST	Baliabide berri bat sortu
PUT	Baliabide bat eguneratu
DELETE	Baliabide bat ezabatu
PATCH	Baliabide bat partzialki eguneratu
HEAD	Goiburukoen eskaera egin

3.1 Taula: HTTP metodoak.

URI-en egitura

RESTful web zerbitzu batek *URI* (***U**niform **R**esource **I**dentifier*) [25] formatu bereziak erabiltzen ditu, erabiltzaileentzako erosoagoak izateko. Oinarrian, *API* bat inplementatzen da eta baliabideak eta bildumak interfaze honen bidez lortzen dira. Baliabideak identifikatzeko sintaxi unibertsal bat erabiltzen da. Honi esker, baliabide bakoitza bere URI-aren bitartez bakarrik atzi daiteke. *URI*-ak argiak eta aurreikusteko modukoak izateak baliabideen atzipena intuitiboa izatea ekartzen du.

URI bat baliabide baten identifikadorea da. Ikus dezagun berriz ere aurreko adibide bat:

```
GET /erabiltzaileak
```

GET metodoa erabiliz, `/erabiltzaileak` baliabidearen eskaera egiten dugu. *REST*-en patroia jarraituz, *URI* horren bidez erabiltzaile guztiak lortuko lirateke, normalean datu-base batean biltegiratuta egongo dira.

Erabiltzaile zehatz bat lortzeko, aldiz, ondorengo *URI*-a erabiliko genuke:

```
GET /erabiltzaileak/erab001
```

Ikus daitekeen moduan, *URI*-ak ulerterrazak eta argiak dira.

API-aren *URI*-ek patroi bat jarraitzen dute normalean, baina garatzaileak bai beste patroi bat jarraitu edota aldaketak egin ditzake. Honako hau da gehienetan erabiltzen diren patroiak:

```
https://domeinua/api/API bertsioa/baliabidea/identifikadorea
https://api.domeinua/API bertsioa/baliabidea/identifikadorea
```

Patroia argiago ikusteko, aztertu dezagun adibide bat. *Twitter*-en erabiltzaile baten jarraitzaileak ikusteko ondorengo *URL*-a erabili dezakegu: `https://api.twitter.com/1.1/followers`. Kasu honetarako *HTTP*-ko *GET* metodoa erabili beharko genuke, informazioa jaso nahi baitugu, jarraitzaile bilduma bat, alegia.

API-aren arabera, eskaera egiten dugunean hainbat parametro gehiago pasa ditzakegu. Esaterako, aurreko adibidearekin jarraituz, sistemari esango genioke jasotako bildumatik hamar jarraitzaile jasotzea besterik ez zaigula interesatzen.

Orokorrean, ideia da *URI*-ak eta beharrezko erregistroak atzitzeko baldintzak ahalik eta argien izatea baliabide eta bildumekin modu errazean lan egiteko.

HTTP egoera-kodeak

REST-en *HTTP* egoera-kodeak erabiltzen dira eskaeren erantzunak arrakastatsuak izan diren edo erroreren bat gertatu den adierazteko. Nahiz eta egoera-kode asko eta desberdinak egon, normalean multzo txiki bat erabiltzen da. Gainera, kode hauek egoeraren arabera zenbakituta daude eta zenbakiaren bidez jakin dezakegu zer arazo gertatu den

eskaera prozesatzerakoan. Eskaeraren erantzuna ondo prozesatu bada, 2xx zenbakiak erabiltzen dira. 3xx familiakoek, berriz, zerbitzariak eskaera prozesatzen buka dezan zerbait falta dela adierazten dute. Bezero aldean erroreren bat gertatu bada, 4xx kodeekin adierazten da. Azkenik, 5xx kode-familia zerbitzari aldeko erroreak adierazteko erabiltzen da. 3.2 taulan gehien erabiltzen diren kodeak agertzen dira, laburbilduta:

Kode-familia	Kodea	Esanahia	Erabilera
2xx	200	OK	Baliabide edo bilduma bat existitzen dela zehazteko.
	201	Created	Baliabide bat sortu dela espezifikatzeko.
	204	No Content	Emaitza egokia errepresentatzeko baina datu itzulerarik gabe.
3xx	304	No Modified	Baliabideak edo bildumak ez duela aldaketarik jasan espezifikatzeko.
4xx	400	Bad Request	Eskaera okerra izan dela espezifikatzeko, hau da, zerbitzariak ezin izan duela prozesatu.
	401	Unauthorized	Eragiketa egin aurretik bezeroa autentikatzea beharrezkoa dela espezifikatzeko.
	404	Not Found	Baliabidea ez dela aurkitu espezifikatzeko.
	405	Method Not Allowed	*API*-ak ez duenean URL-an espezifikatutako metodoa onartzen.
	422	Unprocessable Entity	Eskaera prozesatzeko beharreko parametroak falta direnean edo sintaxia okerra denean.
	429	Too Many Requests	Bezeroak jakin dezan egin dituen eskaera kopuruaren limitea gainditu duela, horretarako politikaren bat existitzen bada.
5xx	500	Internal Server Error	Zerbitzari aldean errore bat gertatu dela adierazteko.
	503	Service Unavailable	Zerbitzaria zerbitzuz kanpo dagoela adierazteko.

3.2 Taula: HTTP egoera-kodeak.

Kode hauei esker, erantzun ulergarriak eraiki daitezke, bezeroak interpreta ditzan. Gainera, garrantzitsua da mezu bidalketan hainbat arau errespetatzea. Adibidez, bezeroak *JSON* formatuan espero badu erantzuna, bidalketaren gorputzean formatu hau erabiltzen ari garela zehaztu beharko dugu.

Garatutako *API*-aren arabera, bidalitako erantzunak zehatzagoak izan daitezke, hau da, bezeroak hauen bidez eskaeraren emaitzari buruz informazio gehiago lor dezake gertatu diren ekintzei buruz. Normalean egiten dena arazoaren deskribapen motz bat erakustea izaten da, erreferentzia kode batekin eta ebazpen posibleren bat. Ondorioz, gure aplikazioan garrantzitsua da gerta daitezkeen errore posibleak identifikatzea eta bakoitzari kode bat esleitzea, gero tratatu ahal izateko.

Ikus dezagun adibide bat. Erabiltzaile bat erregistratzerako orduan hainbat datu eskatzen dituen formulario bat bete behar du, esaterako izena, email-a eta pasahitza. Erabiltzaileak idatzitako datuak egiaztatuko ditu sistemak eta, honen arabera, emaitza aldatuko da. Adibidez, eremuren bat hutsik badago, sistemak errore kode bat eta mezu bat bueltatuko du horren berri emanez. Gauza bera gertatzen da email-aren formatua egokia ez

bada, edota beste arazoren bat gertatzen bada. Dena ondo badago, bezeroak ere jakingo du bere eskaera ondo prozesatu dela. Adibidez, hau izan daiteke *JSON* bidez jasotako errore mezu bat:

```
{
   "errorea":"true",
   "mezua":"Email formatu desegokia"
}
```

HTTP goiburukoak (*HTTP headers*)

RESTful zerbitzuetan goiburukoen erabilera egokia izatea oso garrantzitsua da. Izan ere, eskaerei lotutako konfigurazioak adierazteko beharrezkoak dira. Mota askotako goiburukoak daude, erabileraren arabera banatzen direnak. Alde batetik, *HTTP* eskaeretan erabiltzen direnak daude eta, bestetik, *HTTP* erantzunetan. Nahiz eta goiburuko asko egon, web zerbitzuetan bereziki erabiltzen direnak aztertuko dira 3.3 taulan [26]:

Goiburu izena	Deskribapena	Adibidea
Accept	Erantzunan onartzen diren eduki motak (Content-Types)	Accept: text/plain
Accept-Charset	Onartzen den karaktere-multzoa	Accept-Charset: utf-8
Accept-Encoding	Onartzen diren kodeketen lista acceptable encodings	Accept-Encoding: gzip, deflate
Accept-Language	Erantzunan onartzen diren hizkuntzen lista	Accept-Language: en-CA
Cookie	Zerbitzariak *Set-Cookie* goiburukoaren bidez bidalitako *HTTP cookie*-a	Cookie: $Version=1; Skin=new;
Content-Length	Eskaera gorputzaren luzera 8 *bit*eko *byte*-tan	Content-Length: 348
Date	Mezua bidali zen data eta ordua	Date: Tue, 15 Nov 1994 08:12:31 GMT
Content-Type	Eskaera gorputzaren *MIME* mota	Content-Type: application/x-www-form-urlencoded
Authorization	*HTTP* autentikaziorako kredentzialak	Authorization: Basic bXl1c2VyOm15cGFzcw==

3.3 Taula: HTTP goiburukoen deskribapena.

- ***Content-Type***: goiburuko hau tratatzen ari den fitxategiaren izaera edo formatua *MIME* (***M**ultipurpose **I**nternet **M**ail **E**xtensions*) [27] moten bidez zehazten den goiburukoa da. Protokolo askotan definituta dago. *REST* web zerbitzuetan, askotan *'application/json'* mota erabiltzen da eskaerak egin eta erantzuteko.

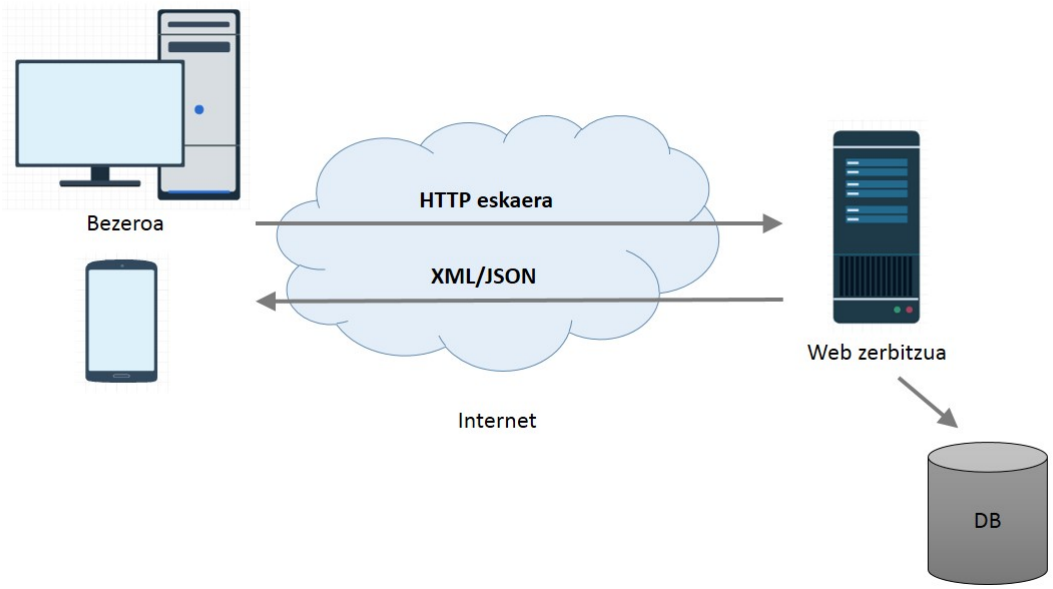
Funtzionamendua

Behin azalduta zer elementuk hartzen duten parte *REST* motako web zerbitzuetan, eskaera bat nola prozesatzen den ikusiko da. 3.2 irudian elkarrekintza hau azaltzen da.

1. Web zerbitzariak *RESTful* web zerbitzu bat izango du exekutatzen. Datu-base batean sistema edo aplikazio bateko datuak izango ditu biltegiratuta.

2. Bezeroak bere PC-tik (edo *smartphone* batetik) *HTTP* eskaera bat bidaltzen dio zerbitzariari Internet bidez. Web zerbitzuak eskaera jaso, prozesatu eta emaitza bueltatuko du, konfiguratuta dagoen formatuan (*XML*, *JSON* edo bestelakoa) eta emaitzarekin batera *HTTP* kodea bidaliko du. Emaitza bueltatzeko, eskaeraren arabera zerbitzariak datu-baseko datuak atzitu beharko ditu.

3. Bezeroak erantzuna jaso eta analizatzen eta interpretatzen du. *HTTP* kodearen arabera jakingo du eskaera arazorik gabe egin den ala ez. Erroreren bat gertatuz gero, kodea ikusita jakin daiteke transmisioaren zer puntuan gertatu den.



3.2 Irudia: RESTful web zerbitzu baten funtzionamendua.

3.1.4 Inplementazio aukerak

API bat garatzeko aukera desberdinak daude. Normalean *framework* bat erabiltzen da *API*-a garatzeko prozesua errazteko. *Framework* bat *software*-a garatzeko azpiegitura antolatu bat da, metodo aurredefinituak eskaintzen dituena. Mota askotako *framework*-ak

daude, lengoaia desberdinetarako. *API*-ak garatzeko lengoaia erabilienak *PHP*, *Python*, *Java*, *Ruby* eta *.NET* dira, beste batzuen artean.

REST web zerbitzuak garatzen laguntzen duten hainbat *framework* daude [28] [29] (*Laravel*, *Slim*, *Silex*, *Phalcon*, etab.), baina *Slim*-en zentratuko gara hurrengo atalan.

Slim

Slim framework-a *PHP*-rekin *RESTful API*-ak sortzeko aukera ematen duen *micro framework*-a da. Oso gomendagarria da batez ere *framework*-ak erabiltzen ikasteko eta aplikazio interesgarriak egin daitezkelako. Abantailak izan arren, desabantailak ere baditu. Izan ere, lehen aipatutako beste *PHP framework*-ekin alderatuz, ez da hain eraginkorra, baina aplikazio txikiagoetarako oso aproposa da.

Funtsean, *Slim*-en eginkizuna *HTTP* eskaerak jaso, dagokion errutina exekutatu eta *HTTP* erantzuna bueltatzen du.

Ezaugarriak

Slim micro-framework-ak abantaila asko dauzka. Egindako eskaerak oso azkar prozesatu ahal ditzake eta, sinplea den arren, beharrezko ezaugarriak eskaintzen ditu. Dagoen dokumentazioa nahiko osatuta dago, web ofizialaz gain, Interneten tutorial eta informazio asko aurki daitezke, bere erabilera oso zabalduta baitago.

Eskaintzen dituen ezaugarrien artean, batzuk aipatuko dira. Alde batetik, *URL* argiak diseinatu eta txantiloiak erabil daitezke. Bestalde, *HTTP Caching* deritzonarako euskarria dauka. Honi esker baliabideak eskaera bakoitzean ez dira deskargatu behar, beharrezko banda zabalera murriztuz. *Cookie* seguruak erabiltzeko aukera ere eskaintzen du. Azkenik, erroreen tratamenduari dagokionez, *log*-ak sortzeko aukera dago eta *debug* modu bat ere dauka.

3.2 Web aplikazioak

Atal honetan web aplikazioak zer diren eta izan dezaketen erabilera azalduko da. Gainera, mota hauetako aplikazioetan erabiltzen diren teknologiak aipatu eta modu laburrean azalduko dira.

Web aplikazioak erabiltzaileei Interneten bidez datuak datu-basera igo eta datu-basetik lortu ahalbidetzen dieten programak dira [30]. Gaur egungo web aplikazioek pertsonen datu sentikorrak jaso, prozesatu, biltegiratu eta transmititzeko aukera ematen dute. Web aplikazioen adibideetako batzuk webgune bat, web bidezko posta elektronikoa, *online* denda edota edukia kudeatzeko sistema dira, beste askoen artean.

Teknikoki, web aplikazioek bezero/zerbitzari eskema jarraitzen dute. Web aplikazioak datu-base bati eskaerak egiten dizkio, *HTTP* protokoloaren bidez, web dokumentuak dinamikoki sortu eta bezeroa zerbitzatzeko. Web dokumentuak *HTML* formatu estandarran sortzen dira, bateragarritasuna bermatzeko.

Nahiz eta web aplikazio mota asko egon, proiektu honetan webguneetan zentratuko dira azalpenak. Webgune bat garatzeko beharreko lehenengo gauza *HTTP* zerbitzari bat da. Erabiliena *Apache* [31] da, kode irekikoa dena. *Apache* zerbitzaria segurua, eraginkorra eta hedagarria da. Gaur egun teknologia eta protokolo asko eta desberdinak erabiltzen dira webguneen orriak garatzeko:

- *PHP*: web-orrietako zerbitzari aldeko kodea inplementatzen duen lengoaia.
- *HTML*: web-orrien edukia osatzen duten elementuei egitura ematea ahalbidetzen duen markaketa lengoaia da.
- *CSS*: web-orrien edukia osatzen duten elementuei itxura ematea ahalbidetzen duen lengoaia da.
- *JavaScript*: web-orrietan elementu dinamikoak izatea ahalbidetzen duen bezero aldeko kodea inplementatzen duen lengoaia. Web nabigatzaileak exekutatzen du.

3.2.1 Datu-base atzipena

PHP bidez datu-baseak atzitzeko hainbat luzapen daude eskuragarri, abstrakzio-geruza bat gehitzen dutenak. Horregatik, garrantzitsua da luzapen bakoitzaren analisi bat egitea, dituen abantailak eta desabantailak zeintzuk diren aztertu. Modu honetan, garatutako aplikaziorako aukera egokiena inplementatzeaz ziurtatuko gara. Atal honetan *MySQL*, *MySQLi* eta *PDO* luzapenak aztertuko dira, batez ere azkeneko biak, gaur egun erabilienak baitira [32] [33].

MySQL luzapena hiruen artean zaharrena da eta duela urte batzuk oso erabilia izan zen. Dena den, batez ere segurtasun arazoak direla eta, zaharkituta dagoela kontsideratzen

da eta gaur egun *MySQLi* bertsio hobetua erabiltzen da. Hortaz, *MySQLi* eta *PDO* aukeren artean egingo da konparaketa.

MySQLi luzapenak *MySQL server* bertsio berrietako ezaugarriak aprobetxatzen ditu, baina bakarrik *MySQL*-rako euskarria dauka. *PDO* (***P**HP **D**ata **O**bjects*) luzapena berriena da eta *API* bat eskaintzen du, onartzen dituen 12 datu-baseak kudeatzeko sistema desberdinekin lan egin ahal izateko, hala nola, *MySQL*, *PostgreSQL*, *SQLite*, etab. Hortaz, *PDO* erabiltzeak abantaila handia dauka, batez ere datu-baseak kudeatzeko sistema etorkizunean aldatzeko asmoa badago. Izan ere, *PDO*-k aldaketa prozesu hau ahalik eta gardenena izatea ahalbidetzen du, kodean lerro gutxi berridatzi behar direlako. *MySQLi* erabiltzen bada, kode osoa berridatzi beharko litzateke.

PDO-k eta *MySQLi*-k objektuei orientatutako (OOP, **O**bjektuei **O**rientatutako **P**rogramazioa) interfaze (*API*) bat eskaintzen dute. *MySQLi*-k, gainera, prozedurazko interfazea ere eskaintzen du, *PDO*-k ez bezala, erabiltzeko errazagoa dena.

Parametroen izendatzeari dagokionez, *PDO* luzapenak eskaintzen duen ezaugarria da, *MySQLi*-k ez bezala. Ezaugarri honek garrantzia dauka sententziei parametroak modu erraz eta malgu batean pasa ahal izateko. Izan ere, parametroen izendatzea erabiliz ez da beharrezkoa parametroen ordena kontuan hartu behar.

Objektuen mapeaketak datu-basetik lortutako datuak objektu moduan tratatzea ahalbidetzen du. Ezaugarri hau bai *MySQLi*-k nola *PDO*-k dute. Honi esker, datu-baseko erregistro bakoitza zuzenean atzi daiteke eta *PHP* objektu bat bezala lortu, bere atributuekin. Adibidez, *User* taula bat *id* eta *name* zutabeekin izanik, *PHP* kodera pasatzean *User* klase bat sortuko genuke, *id* eta *name* atributuekin. Objektuen mapeaketa abstrakzio geruza bat gehitzen du.

Segurtasunari dagokionez, bi luzapenek *SQL injection* erasoari aurre egiteko gai dira, betiere modu egokian egiten badira sententziak (sententzia prestatuak). Sententzia prestatu bat eskaera baten irudikapena da objektu bezala, lehen azaldutako objektuen mapeaketarekin zerikusia duena. Izan ere, sententzia prestatuei esker, kodea sinplifikatzen da eta behin prestatzearekin nahikoa da eskaera berdina hainbat aldiz egin behar bada, parametroen balio desberdinak erabiliz.

Exekuzio abiadurari dagokionez, *MySQLi* azkarragoa da *PDO* baino, baina biak errendimendu ona dute.

Ezaugarri nagusienak aztertuta, esan daiteke oro har *PDO* aukera hobeagoa dela, batez ere etorkizunerako begira kodearen eskalagarritasuna kontuan hartuta. Nahiz eta hasieran

MySQLi baino prestakuntza gehiago eskatu, abantaila gehiago dauzka azken batean.

Aurrekoa laburbilduz, 3.4 taulan [34] azaldutako *MySQLi* eta *PDO* luzapenen arteko desberdintasunak ikus daitezke:

	MySQLi	***PDO***
Datu-base euskarria	*MySQL*	12 datu-baseak kudeatzeko sistema desberdin
API	OOP + prozedurazkoa	OOP
Konexioa	Erraza	Erraza
Parametroen izendatzea	Ez	Bai
Objektu mapeaketa	Bai	Bai
Sententzia prestatuak (bezero aldea)	Ez	Bai
Errendimendua	Azkarra	Azkarra
Biltegiratutako prozedurak	Bai	Bai

3.4 Taula: MySQLi eta PDO luzapenen arteko desberdintasunak.

3.3 *HDF5*

3.3.1 Sarrera

HDF5 (***H****ierarchical* ***D****ata* ***F****ormat* version 5) datuak biltegiratzeko eta kudeatzeko datu-eredu, liburutegi eta fitxategi formatu bat da, kode irekikoa, datu kantitate handiak eta konplexuak biltegiratzeko diseinatuta dagoena. Biltegiratutako datuak mota desberdinetakoak izan daitezke, eta hauen atzipena eta kudeaketa ahalik eta azkarren izatea lortzen da [35] [36] [37].

HDF5-en ezaugarrienetako bat hedagarria eta eramangarria dela da. Sistema eragile nagusietarako eskuragarri dago. *HDF5* bereziki ikertzaileek erabiltzen dute, arlo desberdineko esperimentuen emaitzak biltegiratzeko, baina beste motako erabiltzaileentzat erabilgarria izan daiteke.

Laburbilduz, *HDF5*-en ezaugarri nagusienak hauek dira:

- Bolumen handiko eta/edo datu konplexuak biltegiratzeko nagusiki, baina datu sinpleekin ere lan egin daiteke arazorik gabe.
- Edozein sistema motarako hedagarria.

- Biltegiraketa malgua eta eraginkorra da: Sarrera/Irteera (*Input/Output*) eragiketak.
- *HDF5* erabiltzen duten aplikazioen bilakaera ahalbidetzen du.
- Fitxategiak autodeskribatzaileak dira: erabiltzaileak berak sor ditzake behar dituen metadatuak.

Nahiz eta *HDF5*-k beste formatuen ezaugarri batzuk partekatu, aukera gehiago eskaintzen ditu. *XML*-ren (*eXtensible Markup Language*) antzekoa da, biak baitira autodeskribatzaileak eta datuen arteko dependentziak eta erlazioak definitzea ahalbidetzen dute. Dena den, *XML*-en ez bezala, *HDF5* fitxategietan datu bitarrak gorde daitezke eta edukiak zuzenean atzi daitezke, lehen analizatu behar izan gabe.

Datu-base erlazional batekin alderatuz, *HDF5* objektuak modu naturalago batean adierazi daitezke, direktorio eta fitxategien antzera. Datu-base erlazionalen kasuan taulak erabiltzen dira datuak biltegiratzeko. *HDF5*-en, aldiz, *n* dimentsioko datu-multzoak izan ditzakegu eta bertako elementuak objektu konplexuak izan daitezke.

HDF5 elementu hauetaz dago osaturik:

- **Fitxategi formatu** bat *HDF5* datuak biltegiratzeko.
- **Datu-eredu** bat *HDF5* datuak logikoki antolatzeko eta atzitzeko.
- ***Software*-a** (liburutegiak, lengoaia intergazeak eta erremintak) formatuarekin lan egiteko.

Erabilerari dagokionez, *HDF5* bereziki ikerketa arloan erabiltzen da. Adibidez, klima ikerketarako, urruneko sentsoreen seinaleak jasotzeko, etab. Gainera, gaur egun enpresek ere erabiltzen dute: bioteknologian, banku eta finantzetan, etab.

Atal honetan erabilitako irudi guztiak erreferentzia honetan [37] dauden irudietan oinarritu dira.

3.3.2 Fitxategi formatua

HDF5 fitxategi baten bit-mailako egitura definitzen du. Erabiltzaileek ez dute atal honen inguruko xehetasunik jakin behar.

3.3.3 Datu-eredua

Datu-eredua *HDF5* formatuko fitxategi bateko datuen antolaketa eta espezifikazioa definitzen ditu.

HDF5 datu-eredua osatzen duten bi objektu mota nagusiak **taldeak** (*groups*) eta **datu-multzoak** (*datasets*) dira.

- **Taldea**: direktorio baten antzera lan egiten du. Talde baten barnean metadatuak eta *HDF5* objektuak egon daitezke.
- **Datu-multzoa**: datuak bektore multidimentsionaletan gordetzen dira, datu hauek deskribatzen dituzten hainbat metadatuekin batera.

Taldeek eta datu-multzoek atributu lista bat lotuta eduki dezakete. Atributu batek objektu bati buruzko informazio gehigarria ematen du, erabiltzaileak definituta.

Lehen aipatu moduan, *HDF5*-eko fitxategi egitura fitxategi sistemaren oso antzekoa da. Modu honetan, *root* talde bat izango dugu eta sortzen diren talde edo objektuak talde horren azpian kokatuko dira hierarkian.

Taldeak

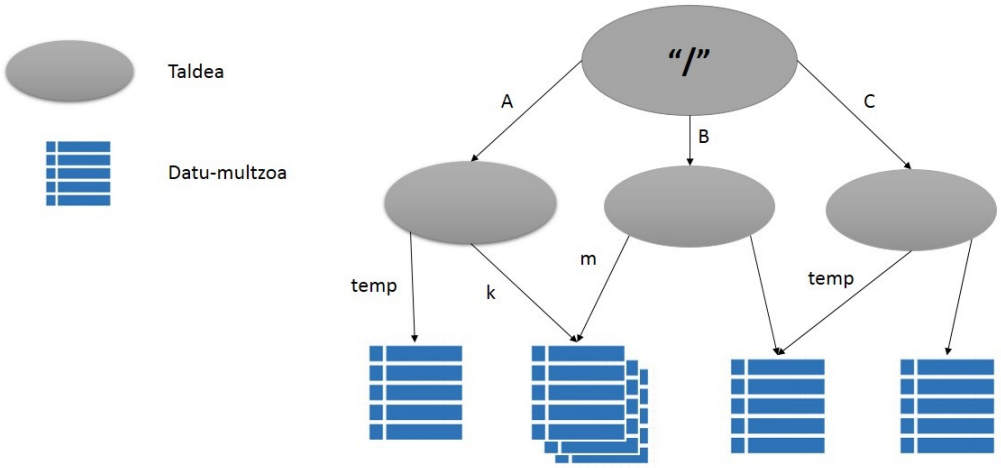
HDF5 taldeek estekak eta objektuak antolatzen dituzte. Edukiontzi bat dela kontsidera daiteke, bertan edozein motako datu-multzoak gordeta daudelarik. *HDF5* fitxategi guztiek erro-talde bat (*root*) daukate, eta honek aldi berean beste azpitalde batzuk izan ditzake.

Azaldutakoa laburbilduz:

- "/": *root* taldea.
- "/t1": *t1* taldea, *root* taldearen azpitaldea da.
- "/t1/t2": *t2* taldea, *t1* taldearen azpitaldea da, eta baita *root* taldearen azpitaldea.

Talde bateko *HDF5* objektuak, datu-multzo bat adibidez, bere talde-*path*-aren bidez dago definituta, 3.3 irudian ikus daitekeen moduan.

Gainera, objektuak parteka daitezke. Kasu horretan, bide desberdinak egongo lirateke objektu bera atzitzeko. 3.3 irudian */A/k* eta */B/m* bideek objektu berdinera apuntatzen dute.

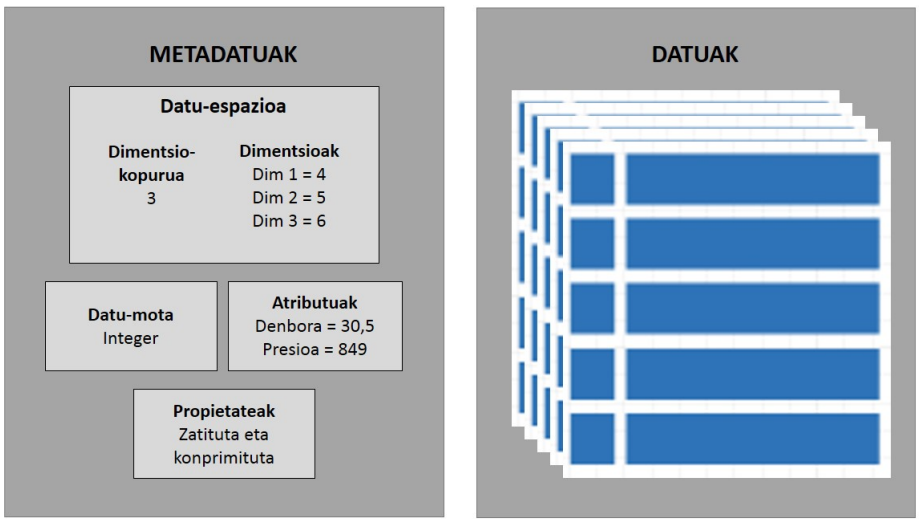


3.3 Irudia: HDF5 taldeak eta datu-ereduak.

Datu-multzoak

HDF5 datu-multzoek datuak antolatzen eta biltegiratzen dituzte. Bi elementuz daude osatuta: metadatuak eta datuak.

Datu-motak, datu-espazioak, propietateak eta atributuak (aukerazkoak direnak) datu-multzo bat deskribatzen dituzten *HDF5* objektuak dira (ikus 3.4 irudia).



3.4 Irudia: HDF5 datu-multzoa.

Datu-motak

Datu-motak datu-multzo bateko elementuak deskribatzen ditu. Hau da, biltegiratutako datuak zer motakoak diren adierazten du.

Bi taldeetan banatzen dira:

1. Datu-mota aurredefinituak: *HDF5*-ek sortu eta kudeatzen ditu. Balio desberdinak izan ditzakete *HDF5* sesio batetik bestera. Bi azpimota bereizten dira:
   - Estandarrak: plataforma guztietan berdinak dira eta *HDF5* fitxategian ikusten direnak dira.
   - Berezkoak (*native*): memoria eragiketak (irakurketa, idazketa) sinplifikatzeko erabiltzen dira eta ez dira berdinak plataforma guztietan.
2. Datu-mota deribatuak: datu-mota aurredefinituetatik eratorriak. Adibidez, karaktere bat baino gehiagoko *string* bat.

Ondorengo datu-motak definitu daitezke *HDF5*-en, beste batzuen artean:

1. *Integer*: osoko zenbakiak.
2. *Float*: zenbaki errealak.
3. *Character*: karaktereak (*byte* 1-eko bektoreak).
4. *Reference*: fitxategiko beste objektu bati erreferentzia.
5. *Enumeration*: balio diskretuen zerrenda.
6. *Compound*: C lengoaiako *struct* antzekoa. Mota honekin taula sinpleak sor daitezke.
7. *User-defined*: erabiltzaileak definituta. Adibidez: 10 biteko osokoak, tamaina finkoko *string*-ak...

Datu-espazioak

Datu-espazioak datu-multzo bateko datu-elementuen egitura deskribatzen du. Elementurik gabekoa (*NULL*), elementu bakarrekoa (eskalarra) edota bektorea izan daiteke.

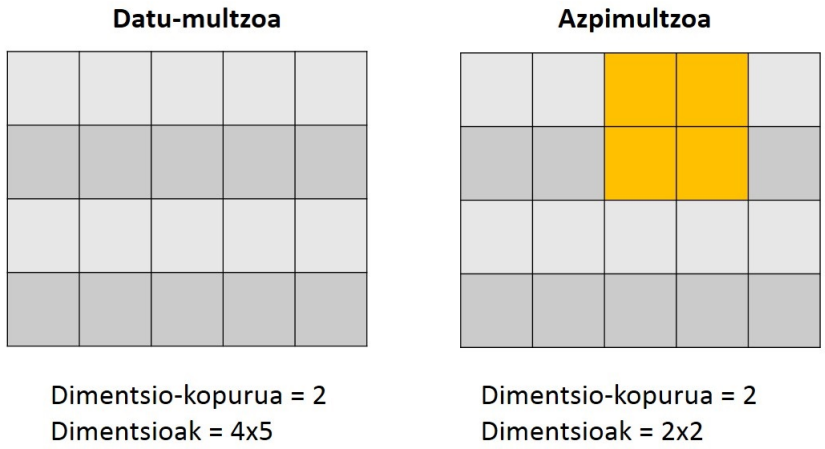
Datu-espazio bat bi elementuz osatuta dago:

1. Dimentsio-kopurua (*rank*).

2. Dimentsioak: finkoak edo mugagabeak izan daitezke.

Datu-espazio batek bi eginkizun betetzen ditu. Alde batetik, datu-multzo baten egitura logikoa definitzen du, hau da, biltegiratuko diren datuek jarraituko duten "eskema" zehazten du. Bestetik, datu-multzo bateko azpimultzoak atzitzea ahalbidetzen du.

3.5 irudian 2 dimentsioko datu-espazio bat ikus dezakegu, 4x6 tamainakoa. 2x2 tamainako azpimultzoa markatuta dago berdez.



3.5 Irudia: HDF5 datu-espazioa.

Propietateak

Propietate bat *HDF5* objektu baten ezaugarri bat da. Normalean erabiltzen diren propietateak defektuz daude definituta, baina hauek alda daitezke *HDF5 API* baten bidez.

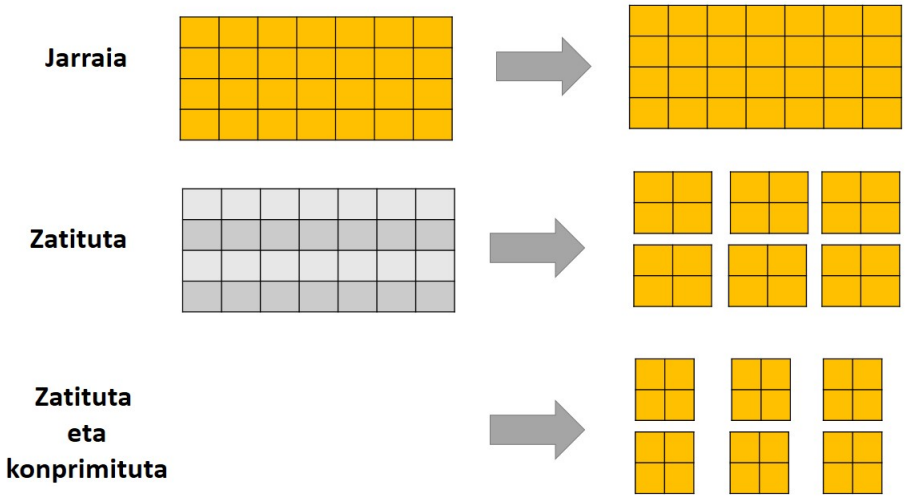
Hainbat propietate daude, baina gehien erabiltzen dena biltegiratze eskemaren propietatea da. Honi esker, datu-elementuak datu-espazioan nola biltegiratuko diren defini daiteke.

Hiru eskema desberdin daude (ikus 3.6 irudia):

- Jarraia (*Contiguous*): datu-elementuak segidan biltegiratzen dira. Defektuz eskema hau jarraitzen da.

- Zatituta (*Chunked*): azpimultzoak azkarrago atzitzea ahalbidetzen du.

- Zatituta eta konprimituta (*Chunked & Compressed*): biltegiraketa eraginkorragoa eta transmisio azkarragoak ahalbidetzen ditu.



3.6 Irudia: HDF5 biltegiratze-eskemak.

Atributuak

Atributuak aukerazkoak diren *HDF5* objektuen ezaugarriak dira, hau da, objektu bati lotuta daude. Bi zatiz osatuta daude: izen bat eta balio bat. Objektu baten atributuak atzitzeko, objektua bera ireki behar da programatzeko orduan, ez baitira objektu independenteak. Normalean tamaina txikikoak dira eta objektu baten metadatuak besterik ez dira.

Atributuek datu-mota bat eta datu-espazio bat dute, beraz, datu-multzo baten antzera lan egiten dute, baina era desberdinean: S/I-ko eragiketak ezin dira egin eta ezta konprimitu edo zabaldu.

3.3.4 *HDF5* softwarea

HDF5 liburutegiak

HDF5 liburutegiak hainbat interfaze edo *API* (***A**pplication **P**rogramming **I**nterface*) dauzka. *API* hauek *HDF5* fitxategiekin lan egiteko funtzio desberdinak dauzkate eta, hauei esker, fitxategien kudeaketa asko errazten da. Funtzio hauek, beste batzuen artean, fitxategiak eta objektuak sortu, irakurri eta idaztea ahalbidetzen digute.

Liburutegia C programazio lengoaian dago idatzita, baina *FORTRAN 90*, *C++*, *Python* eta *Java* lengoaiekin erabiltzeko liburutegiak garatu dira. Dena den, *C* eta *FORTRAN 90* bertsioetarako dokumentazio osoagoa dago. *C* liburutegia aztertuko da memoria honetan.

Liburutegiko funtzio guztiak *H5** aurrizkiarekin hasten dira, non * objektu mota adierazten duen.

API	DESCRIPTION
H5A	***A**ttribute Interface*: atributu atzipena eta kudeaketa.
H5D	***D**ataset Interface*: datu-multzo atzipena eta kudeaketa.
H5F	***F**ile Interface*: fitxategi atzipena eta kudeaketa.
H5G	***G**roup Interface*: talde atzipena eta kudeaketa.
H5L	***L**ink Interface*: loturen kudeaketa.
H5O	***O**bject Interface*: objektuen kudeaketa.
H5P	***P**roperty List Interface*: objektu propietate zerrenden kudeaketa.
H5S	Data**S**pace Interface: datu-espazio definizioa eta atzipena.
H5T	Data**T**ype Interface: datu-mota atzipena eta kudeaketa.

3.5 Taula: HDF5 API-ak.

HDF5 programazioa

Nahiz eta *HDF5 API*-a oso zabala den, kasu gehienetan funtzio gutxi batzuk erabiltzen dira.

HDF5-ekin programatzerako orduan, pauso finko batzuk jarraitzen dira beti:

1. Objektua ireki.
2. Objektua atzitu.
3. Objektua itxi.

Adibidez, *HDF5* fitxategi batekin lan egiterako orduan, lehenbiziko pausoa fitxategia irekitzea da, datu-multzoa ireki baino lehen. Honen arrazoia datu-multzoa irekitzeko funtzioak irekitako fitxategiaren fitxategi-deskribatzailea (*file descriptor*) pasa behar zaiola da. Objektuak irekitzeko ordena garrantzitsua da, beraz.

Bestalde, irekitako objektu guztiak edozein ordenetan itxi daitezke. Behin itxita dagoen objektu bat ezin da atzitu berriro irekia izan arte.

4. KAPITULUA

Proiektuaren garapena

Kapitulu honen helburua proiektuaren garapenaren nondik norakoak azaltzea da. Horretarako, hiru atal nagusi desberdindu dira: sistemaren diseinua, inplementazioa eta egindako balidazioa.

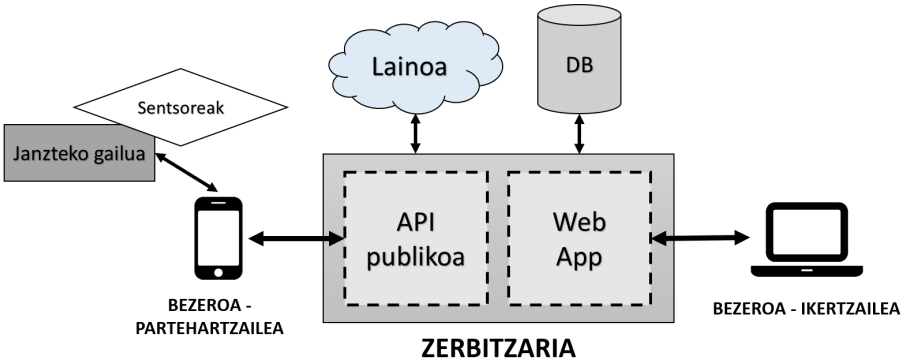
Aurkibidea

4.1 Diseinua

Atal honetan sistemaren diseinuaren nondik norakoak azaltzen dira. Proiektu honen irismena sistema osoa garatzea ez izan arren, sarrera gisa sistema osoaren diseinu orokorra azalduko da. Hurrengo azpiatalan, proiektu honetan ikasleak landutako esperimentuen diseinuaren moduluaren diseinua aurki daiteke.

4.1.1 Sistema osoaren diseinua

4.1 irudian, Ezekiel Sarasua ikaslearekin batera diseinatutako sistemaren eskema orokorra ikusi ahal da:



4.1 Irudia: Sistemaren arkitektura.

Sistemak bezero/zerbitzari arkitektura jarraitzen du eta ondorengo elementuz osatuta dago:

- Bezero aldea:
  - *Android* mugikorra (partehartzailea).
    - Sentsore fisiologikoak.
    - BITalino txartela.
  - Konputagailua (ikertzailea).
- Zerbitzari aldea:
  - Web zerbitzaria.
    - API publikoa.

    * Web aplikazioa.
    * Datu-basea eta biltegiraketa.

Eginkizunen arabera, sistema bi modulu nagusitan banatu da: datuak jasotzeko modulua eta esperimentuen diseinuaren modulua.

Datuak jasotzeko modulua

Modulu honen helburuetako bat partehartzileari mugikortasuna eskaintzea da, *BITalino* txartela eta *Android* mugikor bat erabilita lor daitekeena, hain zuzen ere. Era honetan, partehartzaileak bere datu fisiologikoak eskura ditzake edozein ingurunean.

Lortutako mugikortasuna dela eta, partehartzaileak ez du ikertzailearengana jo beharko esperimenturen bat burutu ahal izateko. Hau da, bi entitateek ez dute ez leku ezta denbora berean konektatuta egon behar.

Izan ere, haririk gabeko konexioak erabili dira. *Android* mugikorra eta txartelaren arteko konexioa *Bluetooth* bidez egingo da. Bestalde, atal honetan geroago moduluen arteko elkarrekintzaren nondik norakoak azalduko dira zehaztasun gehiagorekin.

Ildo berean, eskuratutako datuak ingurune ez-kontrolatu batean lortzen direnez, ez dira laborategi batean eskuratutakoak bezain zehatzak izango, baina bai naturalagoak, ohiko bizimoduan murgilduta.

Azkenik, moduluak duen diseinu gardenari esker, partehartaileak ez du arlo teknikoez arduratu behar, adibidez: *BITalino* txartelaren *MAC* helbidea zehazteaz.

Esperimentuen diseinuaren modulua

Modulu honek hainbat eginkizun betetzen ditu. Zerbitzariaren kudeaketan zentratzen da batez ere, erabiltzaileentzat web aplikazioa eskainiz. Nahiz eta bereziki ikertzaileentzat diseinatuta dagoen, aplikazioaren bidez erabiltzaile motaren arabera ekintza desberdinak egin daitezke:

- Ikertzailea:
    - Sisteman izena ematea.
    - Partehartzaileen erregistroa.

  - Esperimentuen diseinua.

- Ikertzailea eta partehartzailea:

  - Esperimentuen egoera ikusi.
  - Kontuaren kudeaketa.

Erabiltzaile kontuak eta haien konfigurazioak gordetzeko datu-base sistema bat erabiliko da. Bertan ere sortutako esperimentu guztien ezaugarriak eta egoera gordeko dira.

Bestalde, web zerbitzuak (*API* publikoa) datu-basean biltegiratutako datuekin hainbat eragiketa egitea ahalbidetzen du. Hortaz, kanpoko gailuek (*Android* mugikorra), web zerbitzua eskaintzen dituen funtzioak erabilita datuen kudeaketa era garden batean egin dezakete. *Android* aplikazioak funtzio hauek sisteman sartzeko, esperimentuen kudeaketa eta datu fisiologikoen bidalketa egiteko erabiliko ditu.

Esperimentuan eskuratutako datu fisiologikoen kantitatea dela eta, egokia den *HDF5* formatuan biltegiratuko dira.

Moduluen arteko elkarrekintza

Sistemaren funtzionamendu egokirako ezinbestekoa da aipatu berri diren bi moduluen arteko komunikazioa gauzatzea. Horretarako, Ezekiel Sarasua ikaslearekin batera bi moduluen arteko komunikazioa adostu da. Komunikazioa gauzatzeko *HTTP* protokoloa erabiliko da. Azaldutako bi moduluen arteko elkarrekintzak ondorengo prozedura jarraitzen du:

1. Ikertzailea sisteman sartu beharko da, web aplikazioaren bidez, bere kredentzialak erabiliz. Esperimentu bat sortuko du nahi duen partehartzaileari esleituz. Partehartzailea sisteman erregistratuta ez badago, ikertzaileak erregistratzeko aukera izango du.

2. Beste aldetik, partehartailea sisteman bere kontuarekin sartuko da *Android* aplikazioa erabiliz. Zerbitzariari *HTTP POST* eskaera egingo zaio, erabiltzaile izena eta pasahitza bidaliz. Kredentzialak zuzenak izanez gero, zerbitzariak partehartzailearen *API* gakoa eta *BITalino*-ren *MAC* helbidea itzuliko ditu. Datu transmisioak *JSON* formatuan gauzatuko dira.

3. Partehartzaileak atazak eskuratu ahal izateko *Android* aplikazioak *HTTP GET* eskaera egin behar izango du. Horretarako bigarren pausoan jasotako *API* gakoa erabiliko du identifikadore gisa. Atazak *JSON* formatuan jasoko dira.

4. Esperimenturen bat burutu eta gero, eskuratutako datuen *ZIP* fitxategia bidaliko da. Bidalketa *HTTP POST* eskaera baten bidez egingo da, eduki mota (*Content-Type*) *multipart/form-data* dela adieraziko da.

5. Behin *ZIP* fitxategia igo dela, partehartzaileak esperimentua bukatu duela adieraziko du. Esperimentuaren egoera *HTTP PUT* eskaera baten bidez eguneratuko da datu-basean.

6. Jaso berri den *ZIP* fitxategiaren edukiak *HDF5* formatua duen fitxategi batean bihurtuko dira, zerbitzarian biltegiratzen delarik.

7. Ikertzaileak web aplikazioaren bidez esperimentuen informazioa eta egoera edozein momentuan kontsulta dezake, *HDF5* fitxategia deskargatzeko aukera ere izanez.

4.1.2 Esperimentuen diseinuaren moduluaren diseinua

Atal honetan proiektu honetan garatu den moduluaren diseinua azalduko da. Sistemaren egitura eta dituen helburuak kontuan hartuta, inplementatzeko aukera desberdinak komentatuko dira eta azkenean hartutako erabakiak eta hauen zergatiak azalduko dira.

Esperimentuak diseinatzeko aukera egokiena, sistemaren egituraren arabera, web aplikazio baten garapena egitea zen. Beste aukera posibleen artean *Android* aplikazio baten inplementazioa egitea zen, baina ideia hau azkenean datuen eskuraketarako modulurako erabili da. Dena den, etorkizunerako lan bezala, esperimentuen diseinua ere *Android* aplikazioan barneratzea ideia interesgarria izan daiteke.

Datuak eskuratzeko moduluarekin komunikazioa ahalik eta modu errazean ezartzeko aukerak ikusita, egokiena *RESTful* web zerbitzu (*API*) baten inplementazioa dela erabaki da. Izan ere, 3.1.3 atalan azaldu den moduan, *RESTful* web zerbitzu bat sinpleagoa eta arinagoa da, *SOAP* protokoloa ez bezala.

API-a inplementatzeko 3.1.4 atalan azaldu den *Slim framework*-a erabiltzea erabaki da. Beste *framework* batzuen ezaugarriak ikusita eta proiektu honetarako beharrak kontuan hartuz, aukera egokiena *Slim* dela pentsatu da. Izan ere, aurki daitezkeen beste *framework*-ek erabili aurretik ikasteko prozesu luzea behar izaten dute. *Slim*, ordea, erabiltzen ikasteko erraza da, batez ere metodo sinpleak inplementatzeko orduan.

Bestalde, esperimentu adierazgarriagoak egiteko aukera eskaini nahi zaie ikertzaileei. Horretarako, partehartzaile edo subjektu esperimentalek esleitu zaien esperimentua edo ataza burutzen duten bitartean ekintza desberdinak egiten egoteko aukera eman nahi da. Hau lortzeko, "marken" kontzeptua sortu da. Ikertzaile baten ikuspunturako, subjektuak hainbat ekintza desberdin burutzen dituen bitartean bere datu fisiologikoak jaso ahal izatea interesgarria izan daiteke. Esperimentua diseinatzeko orduan, ikertzaileak nahi adina marka edo ekintza definitu ahal izango ditu. Adibide baten bidez azalduko da kontzeptu hau:

- Kirol arloan entrenatzaile bat kirolari baten errendimendua ikusi nahi du, esaterako, korrika egiten ari den bitartean. Dena den, ibilbidean zehar aldapak, geldialdiak edo abiadura aldaketak egon daitezke. Jasotako datu fisiologikoetan aldaketa hauek jasoko lirateke, baina ikertzailearentzat ezinezkoa izango litzateke hauen zergatia jakitea.

  Ondorioz, ekintza edo marka desberdinak definituta, ikertzailearentzat datu fisiologikoak eta antzeman diren aldaketa hauek adierazgarriak izango dira. Hortaz, ikertzaileak honako markak defini ditzake bere esperimentuan, beste batzuen artean:

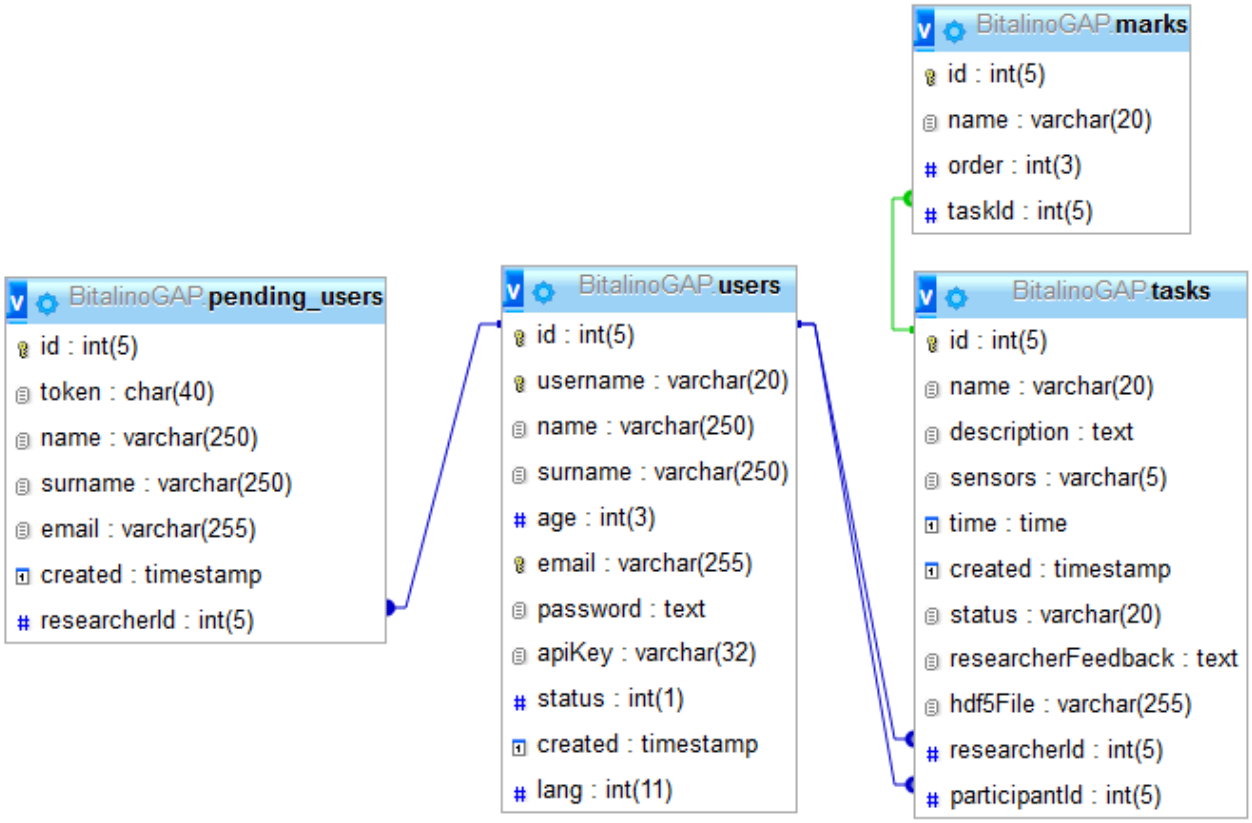
  - Aldapan gora.
  - Aldapan behera.
  - Geldialdia.
  - *Sprint*-a.
  - ...

  Kirolariak, esperimentua burutzen ari den bitartean, *Android* aplikazioaren bidez momentu horretan egiten ari den ekintza aktibatu beharko du.

Hau dena kontuan hartuta, datu fisiologikoen biltegiraketa zehazki nola egin den 4.2.3 atalan azalduko da.

Iraunkortasuna bermatzeko, *MySQL* datu-base bat erabiltzea erabaki da, *phpmyadmin* panelaren bidez kudeatuta. Sistema osoa kontuan hartuta, datu-basean gorde beharreko hainbat elementu identifikatu dira: erabiltzaileak (*users*), erregistratzeko dauden erabiltzaileak (*pending users*), esperimentuak edo atazak (*tasks*) eta denbora-markak edo ekintzak (*marks*).

4.2 irudian diseinatutako taulak eta bere ezaugarriak ikus daitezke. Guztira lau taula sortu dira, lehen azaldutako elementuen arabera:



4.2 Irudia: Datu-basearen diagrama.

- ***users***: sisteman erregistratuko diren erabiltzaile-kontuen informazioa biltegiratuko da. Ondorengo zutabez osatuta dago: identifikadore zenbakia, erabiltzaile izena, izena, abizena, adina, email-a, pasahitza, *API* gakoa, kontuaren egoera, sortzeko data eta hizkuntza.

- ***pending_users***: sisteman erregistro-partziala egin duten erabiltzaileen oinarrizko datuak biltegiratzen dira. Aldi baterako datuak gordeko dira, erabiltzaileak kontua aktibatu arte. Ondorengo zutabez osatuta dago: identifikadore zenbakia, *token*-a, izena, abizena, email-a, sortzeko data eta ikertzailearen identifikadore zenbakia. Ikertzailearen identifikadore zenbakia barne kudeaketak egiteko gordeko da, batez ere.

- ***tasks***: definitzen diren esperimentuak (edo atazak) biltegiratuko dira taula honetan. Ondorengo zutabez osatuta dago: identifikadore zenbakia, izena, deskribapena, sentsoreak, iraupena, sortzeko data, esperimentuaren egoera, ikertzailearen iradokizunak (etorkizunerako lan bezala utzi da), *HDF5* fitxategirako esteka eta ikertzaile eta partehartzaile identifikadore zenbakiak.

- ***marks***: definitutako esperimentuak denbora-markak zehaztuta baditu, taula honetan gordeko dira. Ondorengo zutabez osatuta dago: identifikadore zenbakia, izena, ordena eta esperimentuaren identifikadore zenbakia.

Objektu guztiek identifikadore zenbaki bat dute. Identifikadore hau (*id*) 5 digituko zenbakiak bezala errepresentatzea erabaki da. Objektu motaren arabera, identifikadorearen lehenengo digitua zenbaki desberdina izango da. Modu honetan, *PHP* kodea inplementatzeko orduan errazagoa izango da mota bakoitza identifikatzea. Hortaz, honela definitu dira identifikadore zenbakiak:

- 1xxxx: ikertzailea.
- 2xxxx: partehartzailea.
- 3xxxx: esperimentu edo ataza.
- 4xxxx: marka edo ekintza.

Identifikadore hauek ausaz sortuko dira objektu berri bat sortzean.

Bestalde, zehaztutako erlazioei dagokionez, erabiltzaile bakoitzak (*user*) nahi adina esperimentu edo ataza (*task*) esleituta izan ditzake. Ildo berean, esperimentu bakoitzak nahi adina marka edo ekintza *mark* eduki ditzake lotuta. Bestalde, erregistratzeke dauden erabiltzaileen (*pending users*) informazio partziala gordeko da taulan, erregistroa osatu arte. Orduan taula honetatik ezabatu eta erabiltzaile (*users*) taulan txertatuko da.

4.2 Inplementazioa

Atal honetan sistemaren garapenaren nondik norakoak azaltzen dira, diseinutako moduluaren hiru atal nagusiak nola inplementatu diren azalduz. Erabili den *software*-aren instalazioa eta kodearen zati garrantzitsuenak ere komentatu dira.

4.2.1 Aurrekariak

Ezer inplementatu baino lehen, beharrezko erremintak instalatu beharko dira. Horretarako, web zerbitzua eta aplikazioa garatzeko ordenagailu bat beharko dugu, zerbitzari

gisa lan egiteko, Linux sistemarekin. Kasu honetan *Ubuntu 14.04* instalatuta duen *VirtualBox* makina birtual bat erabiliko dugu. Makina birtuala sortzeko pausoak ikusteko, jo C eranskinera.

Web aplikazioa eta zerbitzua inplementatu ahal izateko *LAMP* instalatuko dugu. *LAMP* (***L**inux, **A**pache, **M**ySQL, **P**HP*) akronimoak web zerbitzariak martxan jartzeko azpiegitura deskribatzen du [38]. Azpiegitura honek konfiguratze prozesua errazten duenez, proiektu honetarako irtenbide egokia da.

1. *Apache* instalatu:

   ```
   sudo apt-get install apache2
   ```

2. *MySQL* instalatu:

   ```
   sudo apt-get install mysql-server
   ```

   MySQL konfiguratzeko, ondorengo komandoa erabiliko dugu. Honen bidez, *MySQL*-rako *root* pasahitza bat aldatu, erabiltzaile kontu anonimoak ezabatu, probak egiteko datu-baseak ezabatu eta beste segurtasun konfigurazioak egin ditzakegu, *MySQL* seguruago izateko.

   ```
   mysql_secure_installation
   ```

   Orain datu-base bat sortuko dugu. Horretarako, *MySQL*-n sartu behar gara root erabiltzailea eta aurreko pausoak esleitu dugun pasahitzarekin.

   ```
   mysql -u root -p
   ```

   MySQL terminala irekiko zaigu eta hasi gaitezke komandoak sartzen. Datu-base bat sortzeko:

   ```
   create database DatuBasea;
   quit
   ```

3. *PHP* instalatu:

   ```
   sudo apt-get install php5 php-pear php5-mysql phpmyadmin
   ```

   Instalatu ondoren, hainbat aldaketa egingo ditugu konfigurazio fitxategian, `/etc/php5/apache2/php.ini`. Ondorengo konfigurazio aukerei esker, errore esanguratsuagoak edukiko ditugu, erroreak ikusi ahal izateko *log* fitxategi bat aktibatuko dugu eta, orokorrean, errendimendua hobetuko dugu.

```
error_reporting = E_COMPILE_ERROR|E_RECOVERABLE_ERROR|
E_ERROR|E_CORE_ERROR
error_log = /var/log/php/error.log
max_input_time = 30
```

Aurreko *log* fitxategia gordetzeko direktorioa sortu behar dugu eta *Apache* erabiltzaileari esleituko diogu:

```
sudo mkdir /var/log/php
sudo chown www-data /var/log/php
```

4. *Apache* konfiguratu:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Apache-ren defektuzko gunearen konfigurazio fitxategian ondorengoa gehituko dugu VirtualHost etiketaren barnean:

```
<VirtualHost>
...

<Directory /var/www/html>
Options Indexes FollowSymLinks
AllowOverride All
Require all granted
</Directory>

</VirtualHost>
```

Amaitzeko, *rewrite* modulua aktibatuko dugu, ondorengo komandoaren bidez:

```
sudo a2enmod
```

Behin oinarrizkoa instalatuta dugula, *Apache* berrabiaraziko dugu konfigurazio berria kontuan hartzeko:

```
sudo service apache2 restart
```

Orain *Slim micro framework*-a instalatuko dugu. Bi modu daude *Slim* instalatzeko: *composer* bidez edo eskuzko instalazioa. Memoria honetan bigarren aukera azalduko da.

Egin beharreko gauza bakarra *Slim GitHub* orritik [39] deskargatu, destrinkotu eta *Slim* direktorioa erabiliko den proiektuaren direktoriora mugitzea da. Proiektu honetan `/var/www/html/libs` direktorioan kopiatu da.

Azkenik, *HDF5* instalatzeko eman beharreko pausoak azalduko dira. Instalatuko den bertsioa 1.8.16 da, baina beste bertsioetarako instalazio-pausoak berdinak izan beharko lirateke.

1. Lehenengo pausoa web gune ofizialetik tar fitxategia deskargatzea da (`http://www.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.16.tar`). Terminal baten bidez fitxategia deskonprimitu eta direktoriora mugituko gara. Instalatzen hasteko komandoa exekutatuko dugu, instalazio direktorioa espezifikatuz.

```
tar xvf hdf5-1.8.16.tar
cd hdf5-1.8.16
./configure --prefix=/usr/local/hdf5
```

2. Bigarren pausoa *make* exekutatzea da. Instalazioa ondo joan den ala ez egiaztatuko dugu ere badaezpada. (Oharra: komandoak errorea ematen badu, *root* baimenekin exekutatuko dugu *sudo* erabiliz: *sudo make*)

```
make
make check                # run test suite
make install
make check-install        # verify installation
```

3. Azkenik, beharrezko liburutegiak eta erremintak instalatuko ditugu:

```
sudo apt-get install hdf5-helpers libhdf5-dev hdf5-tools
```

HDF5 liburutegia erabiltzen dituzten programak konpilatzeko ondorengo komandoa erabiliko dugu.

```
h5cc -o file file.c
```

HDF5 formatuko fitxategiak ikusi ahal izateko ondorengo komandoa erabiliko dugu:

```
h5dump file.h5
```

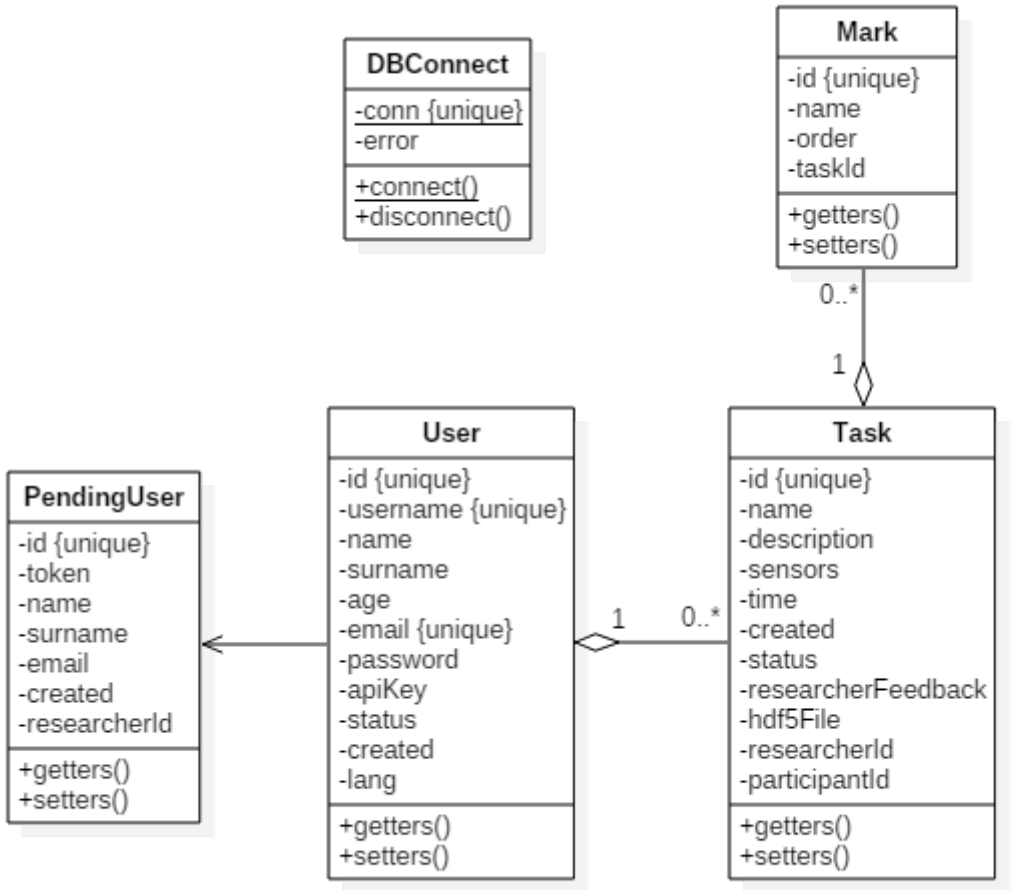
4.2.2 Zerbitzaria

Egitura

Atal honetan zerbitzaria nola prestatu den azalduko da. Lehenik eta behin, datu-baseko informazioa kudeatzeko inplementatu den datu-domeinua azalduko da. Ondoren, fitxategiak kudeatzeko sortu den direktorio egitura azalduko da. Azkenik, zerbitzarira konexio seguruak lortzeko *Apache*-en egindako konfigurazioa aipatuko da.

Datu-eredua

4.3 irudian 4.1.2 atalean aipatutako datu-basearen diagramaren klase diagrama azaltzen da:



4.3 Irudia: Klase diagrama.

Jarraitu den diseinu patroia *MVC* (***M**odel-**V**iew-**C**ontroller*) [40] da. Alde batetik, eredua (*Model*) azaldu diren objektuez (*User, Task, etab.*) osatuta dago, datu-basera konektatzeko klasearekin batera (*DBConnect*). *DBConnect* klasea datu-basea atzitzeko sortu da, *PDO* erabiliz. Konexioa ezartzeko, konfigurazio-fitxategi bat prestatu da, bertan beharrezko datuak daudelarik, hala nola, datu-basearen izena, kredentzialak, etab. Klase honek bi metodo dauzka: *connect* metodo estatikoa, datu-basera konektatzeko eta *disconnect* metodoa, ezarritako konexioa ixteko.

Ereduko objektuak kudeatzeko *Manager* motako klaseak sortu dira. Klase hauen helburu nagusia datu-basea eta objektuen arteko interfaze moduan lan egitea da. Objektu bakoitzeko *Manager* klase bana (*UserManager*, *TaskManager*, *MarkManager* eta *PendingUser*) sortu da, datu-basera instantzia sortu eta *CRUD* eragiketak inplementatzen dutenak. Eragiketa hauez gain, objektu bakoitzerako lagungarriak diren beste metodo batzuk inplementatu dira *Manager*-aren arabera.

Modu honetan, *API*-a eta web aplikazioa inplementatzeko, *Manager* klaseak erabiliko dira datu-baserako transakzio desberdinak burutzeko.

Direktorio egitura

4.4 irudian *API* eta webgunearen garapenerako sortu den direktorio egitura ikus daiteke:

4.4 Irudia: Aplikazioaren direktorio egitura.

API-aren fitxategiak *api* direktorioan gorde dira. Webgunearen fitxategiak, aldiz, beste direktorioetan gorde dira.

Apache konfigurazioa

Azpiatal honetan *Apache* zerbitzarian egin diren konfigurazioak azalduko dira. Ondorengo kodean `/etc/apache2/sites-available/dafiesku.conf` konfigurazio fitxategiaren edukia ikus daiteke:

```
<VirtualHost *:80>
        ServerName dafiesku.com
        ServerAlias dafiesku.com
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
        <Directory /var/www/html>
                Options Indexes FollowSymLinks
                AllowOverride All
                Require all granted
        </Directory>
</VirtualHost>

<VirtualHost *:443>
        # SSL config
        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/certs/server.crt
        SSLCertificateKeyFile /etc/apache2/ssl/private/server.key

        ServerName dafiesku.com
        ServerAlias dafiesku.com
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        <Directory /var/www/html>
                Options Indexes FollowSymLinks
                AllowOverride All
                Require all granted
        </Directory>
</VirtualHost>
```

Webgunea eta web zerbitzua bai *HTTP* nola *HTTPS* bidez atzitzeko konfiguratuko da, nahiz eta transmisioan segurtasuna bermatzeko *HTTPS* erabiliko den.

Orain konfigurazio fitxategian idatzitako lerroak azalduko dira. Bi *VirtualHost* definitu dira, alde batetik *HTTP* protokoloaren bidez atzitu ahal izateko 80 portuan eta, bestetik, *HTTPS* bidez, 443 portuan. Bietan *ServerName* (zerbitzariaren (*host*) izena) eta *DocumentRoot* (erro direktorioa, web-aren bidez atzigarria) berdinak zehaztu dira. Bestalde, zerbitzarian gerta daitezkeen erroreak edo abisuak jasotzeko *log*-en kokapena zehaztu da.

Directory direktiba zehazten den direktorioak aplikatu nahi diren konfigurazio aukerak taldekatzeko erabiltzen da. Kasu honetan hiru zehaztu dira:

- *Options Indexes FollowSymLinks*: *Options* aukera eskuragarri egongo diren ezaugarriak zehazten ditu. Kasu honetan *Indexes* (*index* fitxategien bilaketa automatikoa) eta *FollowSymLinks* (zerbitzariak lotura sinbolikoak jarraitzeko - *API-rako*) interesatzen zaizkigu.
- *AllowOverride All*: *.htaccess* fitxategien erabilera ahalbidetzen du.
- *Require all granted*: atzipena ahalbidetzen du.

.htaccess fitxategia

Aurreko konfigurazioarekin jarraituz, *API*-a `https://domain/api/v1` *URL*-aren bidez atzitzeko *.htaccess* fitxategi bat sortu da `/var/www/html/api/v1/` direktorioan. Fitxategi honen bidez *URL* horretara egindako *HTTP* eskaera guztiak `/var/www/html/api/v1/` direktorioan dagoen *index.php* fitxategira birbidaliko dira.

Birbidalketak ahalbidetzeko *Apache*-k duen *Rewrite* modulua erabili da. Aktibatuta ez badago, terminaletik `sudo a2enmod` komandoa erabiliz egin daiteke. Ondorengo kodeak *.htaccess* fitxategiaren edukia erakusten du:

```
<IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteBase /api/v1/
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteRule ^ index.php [QSA,L]

        SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
</IfModule>
```

Hasteko, direktiba guztiak *IfModule* barnean idatzi dira, hau da, soilik exekutatuko dira *Rewrite* moduluaren fitxategia existitzen bada.

- *RewriteEngine On*: exekuzioan *Rewrite* modulua aktibatzen du.
- *RewriteBase /api/v1/*: berridazketak zer direktorioan egingo diren adierazten du.

- *RewriteCond %{REQUEST FILENAME} !-f*: direktorio horretan existitzen diren fitxategien berridazketak sahiesteko, hau da, fitxategi hauek atzi ahal izateko.

- *RewriteRule îndex.php [QSA,L]*: berridazketa egiteko arauak. Kasu honetan, */api/v1/edozer* idatzita, direktorioan dagoen *index.php* fitxategira birbidaliko du eskaera. Gehitu diren bi *flag*-ak *QSA* (*qsappend*) eta *L* (*qlast*) dira. *QSA*-k eskaeran idatzitako edozein testua *URL* osora eransten du. *L*, aldiz, berridazketa prozesua berehala gelditzeko eta arau gehiago ez aplikatzeko erabiltzen da.

Bestalde, *HTTP*-ko *Authorization* goiburukoa modu erosoan irakurri ahal izateko *API*-an ondorengo lerroa gehitu da:

```
SetEnvIf Authorization "(.*)* HTTP_AUTHORIZATION=$1
```

Lerro honek *PHP* kodetik *Authorization* goiburukoa irakurtzea ahalbidetzen du. Izan ere, *Apache*-ko modulu batzuek segurtasunagatik goiburuko hau desgaitu egiten dute. *SetEnvIf* direktibak eskaeraren arabera ingurune aldagaiei balio desberdinak esleitzea ahalbidetzen du. Kasu honetan, adibidez, *Authorization* goiburukoa erabilita irakurtzen den bigarren parametroa *HTTP_AUTHORIZATION* ingurune aldagaiari esleituko zaio. Honen erabilera aurrerago azalduko da zehaztasun gehiagorekin, web zerbitzuaren atalan.

Ziurtagiriak prestatu

Webgunea *HTTPS* bidez atzitu ahal izateko, zerbitzariak ziurtagiri bat behar du autentikaziorako.

Ziurtagiria sortzeko, *openssl* tresna erabili da. Instalatzeko, `sudo apt-get install openssl` komandoa exekutatuko dugu terminalean.

openssl erabiliz, ziurtagiri autosinatu bat sortuko dugu. Ziurtagiri autosinatuak ez daude ziurtagiri autoritate (*CA, **C**ertification **A**uthority*) nagusi edo komertzial baten bidez balidatuak. Ondorioz, errealitatean ezingo genituzke erabiliko, baina bai garapen fasean probak egiteko. Ziurtagiri autosinatuak erabiliz, bezero eta zerbitzari arteko komunikazioa zifratua egongo da, baina zerbitzariaren egiazkotasuna modu desberdinean egiten da.

Ziurtagiri autoritateak igorritako ziurtagiri batean, autoritateak berak balidatzen du zerbitzariaren identitatea. Izan ere, zerbitzariaren ziurtagiri eskaera ziurtagiri autoritateak

sinatzen du. Modu honetan, gune bateko zerbitzariak duen ziurtagiria nabigatzaileak biltegiratzen duen ziurtagiriarekin bat egiten du, eta, ondorioz, konfiantzazkoa dela erabakitzen du.

Hauek dira jarraitu beharreko pausoak:

1. Zerbitzariaren ziurtagiria (*server.crt* fitxategia) gordetzeko direktorio bat sortuko dugu:

```
sudo mkdir /etc/apache2/ssl/
sudo mkdir /etc/apache2/ssl/certs
```

2. Beste direktorio bat sortuko dugu gako pribatua (*server.key* fitxategia) gordetzeko:

```
sudo mkdir /etc/apache2/ssl/private
```

3. Zerbitzariaren gako pribatua sortuko dugu:

```
openssl genrsa -out server.key 1024
```

4. Zerbitzariaren ziurtagirirako sinadura eskaera (*server.csr* fitxategia) sortzen dugu:

```
openssl req -new -key server.key -out server.csr
```

   Hainbat datu eskatuko ditu, garrantzitsuena *Common name* edo domeinuaren izena izanik. Honetan *Apache*-ko gunearen *ServerName* izenaren berdina idatzi beharko dugu, kasu honetan *dafiesku.com*.

5. Zerbitzariaren ziurtagiria sinatzen dugu:

```
openssl x509 -req -days 365 -in server.csr -signkey
server.key -out server.crt
```

6. Zerbitzariaren ziurtagiria sinatzeko eskaeraren fitxategia ezabatzen dugu:

```
rm server.csr
```

7. Behin ziurtagiri autosinatua lortu dugula, zerbitzariaren izena *hosts* fitxategian gehituko dugu:

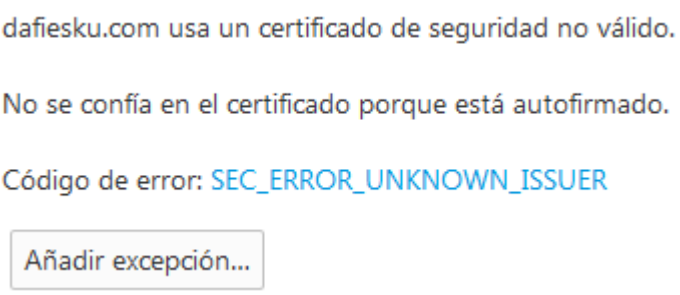
```
127.0.0.1       localhost
127.0.1.1       user-VirtualBox
127.0.0.1       dafiesku.com


# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
```

Lehen azaldutako *Apache* konfigurazio fitxategira bueltatuz, ondorengo hiru lerro hauek gehitu dira *HTTPS* atzipena baimentzeko:

```
# SSL config
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/certs/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/private/server.key
```

Apache-ko *SSL* modulua aktibatuko dugu, eta lehen sortutako ziurtagiri sinatua eta gako pribatuaren kokapenak zehaztuko ditugu.

Guzti honekin bukatuko genuke pauso hau. Dena den, gaur egungo nabigatzaileek, segurtasun arrazoiengatik, ez dituzte ziurtagiri autosinatuak onartzen (ez dagoelako balidatuta kanpoko entitate baten bidez) eta 4.5 irudiko errorea agertuko da.



4.5 Irudia: Nabigatzaileak itzulitako errorea, ziurtagiri autosinatu bat erabiltzean.

Kasu honetan probak egiten ari garenez, nabigatzailean salbuespena gehitu dezakegu “Añadir excepción” botoia sakatuz.

Web zerbitzua

Atal honetan web zerbitzuaren (*API*) inplementazioa nola egin den azalduko da. *REST API*-a inplementatzeko *Slim framework*-a erabili da. Nahiz eta idatzitako kode guztia ez azaldu, *Slim* nola erabili den eta *HTTP* metodo desberdinak erabiltzen dituzten metodoak nola inplementatu diren azalduko da.

API-aren kode guztia fitxategi batean (*index.php*) idatzi da, `/var/www/html/api/v1/` direktorioan kokatuta dagoena. Eskaera guztiak *JSON* formatuan egin dira, bai eskaerak jasotzeko eta erantzunak bidaltzeko.

Fitxategi honen nondik norakoak azalduko dira orain. Hasierako pausoa beharrezko liburutegiak eta fitxategiak inportatzea da. Kasu honetan, *Slim* eta datu-ereduko objektuak tratatzeko *Manager* klaseak inportatuko ditugu. Ondoren, *Slim* hasieratu eta instantzia bat sortuko dugu. `$app` aldagaiaren bidez kontrolatuko ditugu *Slim*-ek eskainitako funtzioak. *Manager* eta klase laguntzailea (*Helper*) hasieratuko ditugu, bakoitzaren instantziak sortuz. Puntu honetan beharrezko *HTTP* eskaerak tratatzeko metodoak inplementatuko ditugu. Azkenik, fitxategiaren bukaeran, *API*-a gaitzeko `$app->run()` metodoa exekutatuko dugu.

```
<?php
require $_SERVER[DOCUMENT_ROOT].'/libs/Slim/Slim.php';

/* Initialize Slim lib. */
\Slim\Slim::registerAutoloader();
$app = new \Slim\Slim();

/* Initialize DB Managers. */
$um = new UserManager();
$tm = new TaskManager();
$mm = new MarkManager();

/* Initialize Helper. */
$h = new Helper();

/* API methods */
...
```

```
/* Run app */
$app->run();
```

Puntu honetan has gaitezke *API*-ak kudeatuko dituen *HTTP* eragiketa desberdinak definitzen. Dena den, metodo hauen inplementazioa errazteko, errepikatuko diren hainbat ekintza biltzen dituzten funtzioak inplementatu dira. Alde batetik, `verifyRequiredParams()` funtzioa, eskaeretan jasotako parametroak egiaztatzeko eta, bestetik, `echoRespnse()` *HTTP* erantzunak formatu egokian sortu eta bidaltzeko.

Funtzio honek eskaera metodoak behar dituen *JSON* eremu guztiak pasa diren eta ondo idatzita dauden egiaztatzen du. Parametro bezala bektore (*array*) bat hartzen du, non bere elementuak eskaera prozesatzeko beharrezko *JSON* eremuak diren. Honen erabilera argitzeko, ikus atal honetan aurrerago azalduko diren inplementatutako hainbat metodo.

```
<?php
/**
 * Verify required parameters for the request.
 * @param array $required_fields
 */

function verifyRequiredParams($requiredFields)
{
        /* Variables */
        ...

        /* Get Slim instance */
        $app = \Slim\Slim::getInstance();

        /* Read 'Content-Type' header */
        $contentType = $app->request->headers->get('Content-Type');

        if ($contentType == 'application/json') {
                $json = $app->request->getBody();
                $requestParams = json_decode($json, true);
        } else {
```

```
            ...
        }

        /* Check the requested params and save the error fields. */
                foreach ($requiredFields as $field) {
                        if (!isset($requestParams[$field]) ||
                        strlen(trim($requestParams[$field])) <= 0) {
                                $error = true;
                                $errorFields .= $field.', ';
                        }
                }

        /* Required params are empty or not given. */
        /* Send error message and stop the Slim instance. */
        if ($error) {
                $response = array();
                $response['error'] = true;
                $response['text'] = 'Required param(s) '.
                substr($errorFields, 0, -2).' not given or empty.';
                echoRespnse(400, $response);
                $app->stop();
        }
}
?>
```

Beharrezko aldagaiak definitu eta gero, *Slim* instantzia eskuratuko dugu. Ondoren, bezeroak bere eskaeran *Content-Type* goiburukoa mota egokian idatzi duela (’*application/json*’) egiaztatuko dugu. Hala bada, *JSON* gorputza deskodetu eta \$requestParams bektorean (*key, value*) eran gordeko dira irakurritako balioak. Hurrengo pausoa bektore honetan dauden *JSON* eremuak (*key*) funtzioari parametro bezala pasatako bektorean (\$requiredFields) daudenekin konparatuko ditugu. Gaizki daudenak edo falta direnak *\$errorFields* bektorean gordeko dira. Erroreren bat gertatu bada, ondorengo erantzuna, *JSON* formatuan, itzuliko zaio bezeroari, 400 (HTTP_OK) *HTTP* kodearekin:

```
        {
```

```
            "error":"true",
            "text":"Required param(s) x, y not given or empty"
        }
```

Errorea gertatu denez, *Slim* instantzia gelditu dugu `$app->stop()` funtzioaren bidez.

Orain erantzunak bidaltzeko inplementatutako funtzioa azalduko da, laburki. Funtzio honek bi parametro hartzen ditu: *$statusCode*, *HTTP* egoera kodea adierazten duena, eta *$response* bektorea, *JSON* formatuan bidaliko den erantzuna.

```
<?php
/**
 * Send json response.
 *
 * @param string $statusCode HTTP status code
 * @param int    $response   json response
 */
function echoRespnse($statusCode, $response)
{
        /* Get Slim instance */
        $app = \Slim\Slim::getInstance();

        /* HTTP status code. */
        $app->status($statusCode);

        /* Response content type: json. */
        $app->contentType('application/json');
        echo json_encode($response);
}
?>
```

Lehen bezala, *Slim* instantzia lortuko dugu. Parametro bezala pasatako *HTTP* egoera kodea eta formatu mota (*Content-Type*) esleituko diogu erantzunari. Azkenik, parametro bezala pasatako bektorea *JSON* formatuan kodetuko da, *json_encode* funtzioaren laguntzaz.

Funtzio laguntzaile hauek azalduta, hurrengo pausoa eskaera desberdinak nola inplementatu diren azaltzea da. Kodea nahiko luzea denez, eskaera mota bakoitzaren inplementazioaren zati garrantzitsuenak azalduko dira.

GET eskaera

GET eskaera azaltzeko, erabiltzaile baten atazak edo esperimentuak eskuratzeko inplementatu den metodoa azalduko da. Metodoa atzitzeko *URI*-a */api/v1/tasks* izango da. Metodoa definitzeko, *get()* idatziko dugu eta izena (*tasks*) pasako diogu lehenengo parametro bezala. Bigarren parametroa metodo honek zer egingo duen adierazten duen funtzioa izango da. Funtzio honi barnean erabiliko diren aldagaiak adierazi behar zaizkio. Kasu honetan *Slim* instantzia aldagaia eta lehen hasieratu diren *Manager* eta *Helper* klaseen instantziak pasako dizkiogu.

Kasu honetan bi goiburuko irakurriko ditugu: *Content-Type*, lehen bezala, eta *Authorization*. Bezeroak, *Authorization* goiburu hau erabilita, erabiltzaile kontuaren *API* gakoa bidaliko du, autentikazio modu gisa. Modu honetan, atzipena mugatzen da. Alde batetik, erabiltzaile baten esperimentuak *API*-aren bidez lortzeko, *API* gakoa izatea beharrezkoa da eta, bestetik, bakarrik erabiltzaileak berak ikus ditzake bere esperimentuak.

```php
<?php
/**
 * Get user tasks.
 */
$app->get(
        '/tasks',
        function () use ($app, $um, $tm, $mm, $h) {

                /* Get headers */
                $contentType = $app->request->headers->get('Content-Type');
                $req = $app->request->headers->get('Authorization');

                ...
?>
```

Hurrengo pausoa *API* gakoa egiaztatzea da, hau da, pasatako gakoa sisteman erregis-

tratuta dagoen ala ez. Bestalde, *Manager* klase desberdinak erabiliz, bektore bat sortuko dugu, esperimentu bakoitzaren informazioa eta dituen markak edo ekintzekin batera.

```
<?php
        $valid = $um->isValidApiKey($apiKey);
        if ($valid) {
                $user = $um->getByApiKey($apiKey);
                $tasks = $tm->getByUser($user->getId());
                foreach ($tasks as $t) {
                        $ta = $h->obj2array($t);
                        $ta['marks'] = array();
                        $marks = $mm->getByTaskId($t->getId());
                        foreach ($marks as $m) {
                                $r = $h->obj2array($m);
                                $ta['marks'][] = $r;
                        }
                        $res[] = $ta;
                }
?>
```

Behin bektorea sortu dugula, erantzuna bidaliko dugu. Erroreak egon badira ere (*API* gako okerra, adibidez), 200 *HTTP* egoera kodea (*HTTP_OK*) bidaliko dugu, bezeroak bere eskaera prozesatu dela jakin dezan.

```
<?php
        echoRespnse(200, $response);
?>
```

POST eskaera

POST eskaera azaltzeko, erabiltzaile bat sisteman sartzeko (*login* egiteko) inplementatu den metodoa azalduko da. Metodoa atzitzeko *URI*-a */api/v1/login* izango da. Metodoa definitzeko, *post()* idatziko dugu eta izena (*login*) pasako diogu lehenengo parametro bezala. Bigarren parametroan *Slim* instantzia aldagaia eta lehen hasieratu diren *Manager* eta *Helper* klaseen instantziak pasako dizkiogu.

Content-Type goiburukoa irakurriko dugu. Kasu honetan bezeroak *JSON* formatuan

bidali beharrezko parametroak *loginName*, hau da, erabiltzaile izena edo email-a, eta *password* pasahitza dira. Adibidez:

```
{
        "loginName":"user01",
        "password":"1234"
}
```

Ondoren, *JSON* parametroak eta eduki mota ondo badaude, kredentzialak egokiak diren egiaztatuko da. Egokiak badira, erabiltzailearen datuak itzuliko dira, *JSON* formatuan, *BITalino* txartelaren *MAC* helbidearekin batera. Bestela, errorearen berri emango zaio bezeroari.

```
<?php
/**
 * User login.
 */
$app->post(
'/login',
function () use ($app, $um, $h) {

        $contentType = $app->request->headers->get('Content-Type');
        $response = array();

        /* Verify parameters. */
        verifyRequiredParams(array('loginName', 'password'));

        /* POST request with json format. */
        if ($contentType == 'application/json') {
                $json = $app->request->getBody();
                $data = json_decode($json, true);
        } else {
                ...
        }
```

```
        /* Check username/email and password. */
        if ($um->checkLogin($data['loginName'], $data['password'])) {
                /* Get user from the database. */
                $user = $um->getByLoginName($data['loginName']);
                $response = $h->obj2array($user);
                $response['bitalinoMAC'] = '98:D3:31:B2:14:90';
        } else {
                /* Wrong credentials. */
                $response['error'] = true;
                $response['text'] = 'Wrong credentials.';
        }

        echoRespnse(200, $response);
});
?>
```

PUT eskaera

PUT eskaera azaltzeko, ataza edo esperimentu zehatz bat eguneratzeko inplementatu den metodoa azalduko da. Metodoa atzitzeko *URI*-a */api/v1/tasks/id* izango da, parametro bat duena (*id*, eguneratu nahi den esperimentuaren identifikadore zenbakia). Metodoa definitzeko, *put()* idatziko dugu eta izena (*/tasks/:id*) pasako diogu lehenengo parametro bezala. Bigarren parametroan *Slim* instantzia aldagaia eta lehen hasieratu diren *Manager* eta *Helper* klaseen instantziak pasako dizkiogu. Kasu honetan parametro bat dugunez, *id* balioa funtzioari pasako diogu `function ($id)`.

```
<?php
/**
 * Update task.
 * @param $id
 */

$app->put(
'/tasks/:id',
function ($id) use ($app, $um, $tm, $h) {
```

```
        /* Get headers */
        $contentType = $app->request->headers->get('Content-Type');
        $req = $app->request->headers->get('Authorization');

                ...
?>
```

Honetan ere, *API* gakoa balidatuko dugu, *GET* eskaeran egin den moduan. Esperimentuaren informazioa eskuratuko dugu eta, identifikadore zenbakia duen esperimenturen bat existitzen bada, orduan *PUT* eskaera prozesatuko da. *JSON* formatuan eguneratu nahi diren esperimentuaren atributuak pasako behar ditu bezeroak.

```
<?php
        if ($id != '') {
                $valid = $um->isValidApiKey($apiKey);
                if ($valid) {
                        $t = $tm->get($id);
                        if ($t != null) {

                                /* PUT request with json format. */
                                if ($contentType == 'application/json') {
                                        $json = $app->request->getBody();
                                        $data = json_decode($json, true);

                                ...
                                }
?>
```

Pasa diren parametroen arabera, esperimentuaren objektuan aldaketak egingo dira. Bestalde, existitzen ez den atributuren bat aldatu nahi bada, *$errorValues* bektorean gordeko da, gero bezeroari erantzunarekin batera bidaltzeko.

```
<?php
        ...
        /* Read PUT params. */
```

```
        foreach ($data as $key => $value) {
                switch ($key) {
                        case 'name':
                                $t->setName($data['name']);
                                break;
                        case 'description':
                                $t->setDescription($data['description']);
                                break;
                        case 'sensors':
                                $t->setSensors($data['sensors']);
                                break;
                        case 'time':
                                $t->setTime($data['time']);
                                break;
                        case 'status':
                                $t->setStatus($data['status']);
                                break;
                        case 'hdf5File':
                                $t->setHdf5File($data['hdf5File']);
                                break;
                        default:
                                $errorValues[$key] = $data[$key];
                                break;
                }
        }
        ...
);
?>
```

Web aplikazioa

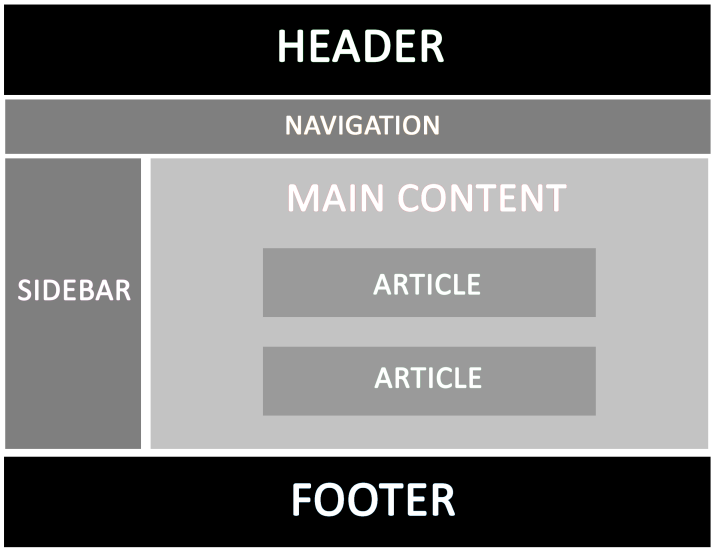
Atal honetan web aplikazioaren (webgunea) garapenaren nondik norakoak azalduko dira. Lehenik eta behin, interfazea azalduko da eta, ondoren, interfazearen kodearen atal garrantzitsuenak komentatuko dira.

Egitura nagusia

Lehenik eta behin, webgunea inplementatzeko jarraitu den egitura nagusia azalduko da. Horretarako, *HTML5* lengoaia [41] erabili da, webguneak egituratzeko elementu berri eta lagungarriak dituena. Webguneari itxura emateko *CSS* [42] lengoaia erabili da eta zerbitzari aldeko programazioa *PHP* lengoaia baliatuz burutu da.

4.6 irudian [43], jarraitu den webgunearen egitura ikus daiteke, webguneek duten ohiko egitura estandarra dena. Alde batetik, goiburuko bat sortu da, bertan sistemaren logoa eta goiko ezkerreko aldean hizkuntza aukeratzeko estekak gehitu dira. Goiburukoa *HTML5*-ek duen <*header*> etiketaren bidez definitu da. Beste aldetik, orri-oin bat ere sortu da, <*footer*> etiketa erabiliz.

Header-a eta *footer*-aren artean dagoen "gorputza" dinamikoa izango da. Bertan <*section*> etiketa eta honen barnean <*article*> etiketa erabiliko dira. *Header*-a eta *footer*-a estatikoak izango dira. Hortaz, aldatzen den edukia gorputza izango da, *jQuery* [44] bidez aldatuko dena.



4.6 Irudia: Webgunearen HTML5 egitura.

Web aplikazioak bi pantaila edo orri nagusi ditu. Alde batetik, hasierako orria, bertan erregistroa egiteko formularioa eta sisteman sartzeko beste formulario bat agertzen dira. Bestetik, sisteman sartuta erabiltzaile orria, erabiltzaile motaren arabera hainbat aukera desberdin daudelarik.

Hizkuntza aldaketa

Webgunea hiru hizkuntzetara itzuli da: euskara, gaztelera eta ingelesa. Hizkuntza aukeratzeko menua goiburuko ezkerreko aldean kokatu da. Funtzionalitate hau inplementatzeko, 4.4 irudian ikus daitekeen *lang* direktorioan hainbat fitxategi sortu dira. Alde batetik, hizkuntza bakoitzera itzulpenak egin ahal izateko fitxategi bana. Fitxategi bakoitzean *define()* funtzioa erabiliz, izen bera duten aldagaiak definitu dira, balioa (itzulpena) aldatuz hizkuntzaren arabera. Adibidez:

```
<?php
        // Euskaraz (lang_eu.php)
        define('USERNAME', "Erabiltzaile izena");
        define('NAME', "Izena");
        define('SURNAME', "Abizenak");
        ...
        // Gazteleraz (lang_es.php)
        define('USERNAME', "Nombre de usuario");
        define('NAME', "Nombre");
        define('SURNAME', "Apellidos");
        ...
        // Ingelesez (lang_en.php)
        define('USERNAME', "Username");
        define('NAME', "Name");
        define('SURNAME', "Surname");
?>
```

Orain hizkuntza aldaketa nola egiten den azalduko da. Horretarako, goiburuan *HTML* menu bat (<*nav*> etiketa) gehitu da, hiru estekekin, *switchLang.php* fitxategira lotura dutenak. Hizkuntzaren arabera, *lang* aldagaiari zenbaki bat esleitu zaio.

```
<nav id="lang-menu">
   <ul>
      <li>
         <a href="/../lang/switchLang.php?lang=1">English</a>
      </li>
```

```
        <li>
            <a href="/../lang/switchLang.php?lang=2">Español</a>
        </li>
        <li>
            <a href="/../lang/switchLang.php?lang=3">Euskera</a>
        </li>
    </ul>
</nav> <!-- / nav -->
```

switchLang.php fitxategian, saioa hasieratuko dugu. Oraingo *URL* gordeko dugu *$goback* aldagaian, *$_SERVER['HTTP_REFERER'] PHP* aldagaiaren bidez lortuko duguna. Ondoren, *GET* metodoaren bidez, lehen aipatutako *lang* aldagaiak duen balioa irakurriko dugu eta saio aldagai batean gordeko dugu. Ondoren, *URL* berera egingo da birbidalketa, aukeratutako hizkuntza kargatuz.

```
<?php
    // switchLang.php
    session_start();

    // Get current url
    $goback = $_SERVER['HTTP_REFERER'];
    $lang = $_GET['lang'];
    $_SESSION['lang'] = $lang;

    // Go to current url
    header("location: $goback");
?>
```

defineLang.php fitxategia aukeratzen den hizkuntzari dagokion fitxategia kargatzeaz arduratzen da. Defektuz hizkuntzarik aukeratuta ez badago, *lang* saio aldagaiari 1 balioa (ingelesa) esleituko diogu. Fitxategi hau webgunearen goiburukoan (*header.php* fitxategian) txertatu beharko dugu, edozein orritan hizkuntza alda dezan.

```
<?php
    // defineLang.php
    session_start();
```

```
    if (!isset($_SESSION['lang'])) {
        $_SESSION['lang'] = 1;
    }

    // English (default)
    if ($_SESSION['lang'] == 1) {
        include_once $_SERVER['DOCUMENT_ROOT'].'/lang/lang_en.php';
    // Spanish
    } elseif ($_SESSION['lang'] == 2) {
        include_once $_SERVER['DOCUMENT_ROOT'].'/lang/lang_es.php';
    // Basque
    } elseif ($_SESSION['lang'] == 3) {
        include_once $_SERVER['DOCUMENT_ROOT'].'/lang/lang_eu.php';
    }
?>
```

Hasierako orria

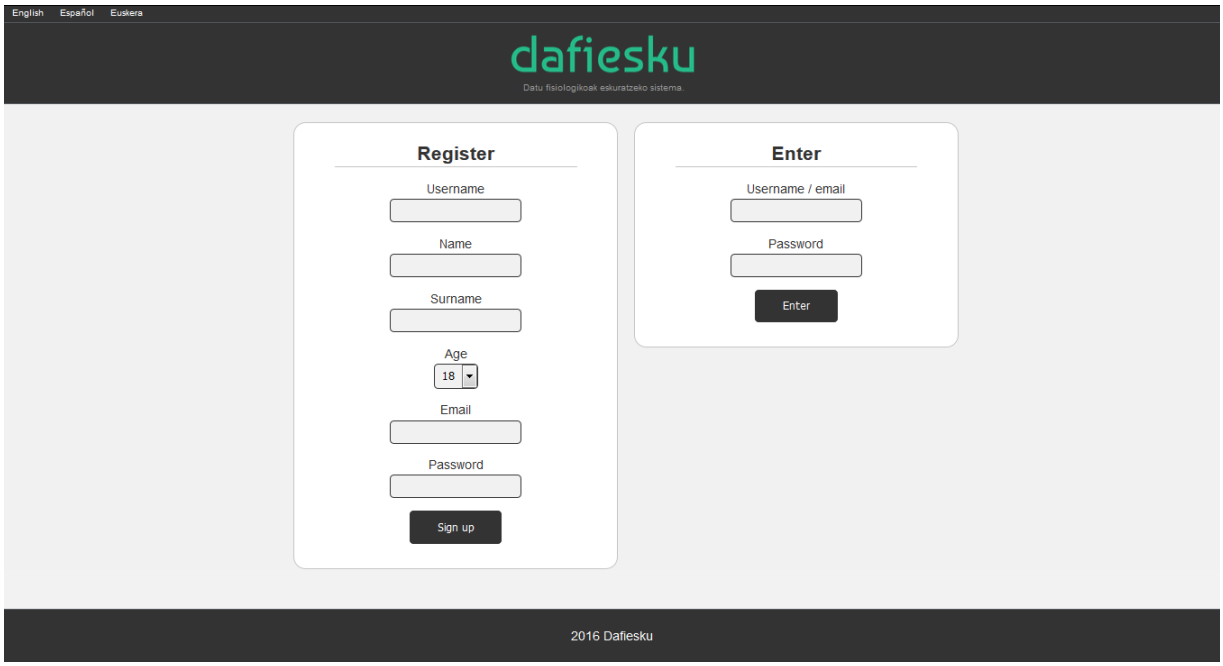
Webgunearen hasierako orriak lehen azaldutako egitura jarraitzen du (ikus 4.7 irudia). Web-orriaren gorputzean bi formulario txertatu dira. Alde batetik, erregistratzeko formularioa, *register.php* fitxategian idatzi dena, eta, bestetik, sistemara sartzeko formularioa, *login.php* fitxategian idatzita. Hasierako orriaren elementu guztiak erro direktorioan kokatutako *index.php* fitxategian zehaztu dira.

Erregistro prozesua azalduko da orain. Bi erabiltzaile mota desberdintzen dira, alde batetik ikertzaileak eta, bestetik, partehartzaileak edo subjektu esperimentalak. Ikertzaileek sisteman erregistroa egin dezakete zuzenean, subjektu esperimentalek ez bezala. Partehartzaileen erregistro prozesua atal honetan aurrerago azalduko da zehaztasunarekin.

Ikertzaileak formularioa bete eta gero, erregistratutako email helbidera mezu bat bidaliko zaio kontua aktiba dezan. Aktibatu arte, ikertzailea ezingo da sisteman sartu.

Erabiltzaile orria

Orri hau erabiltzailea sisteman sartu denean ikusiko duena da (ikus 4.8 irudia). Ha-

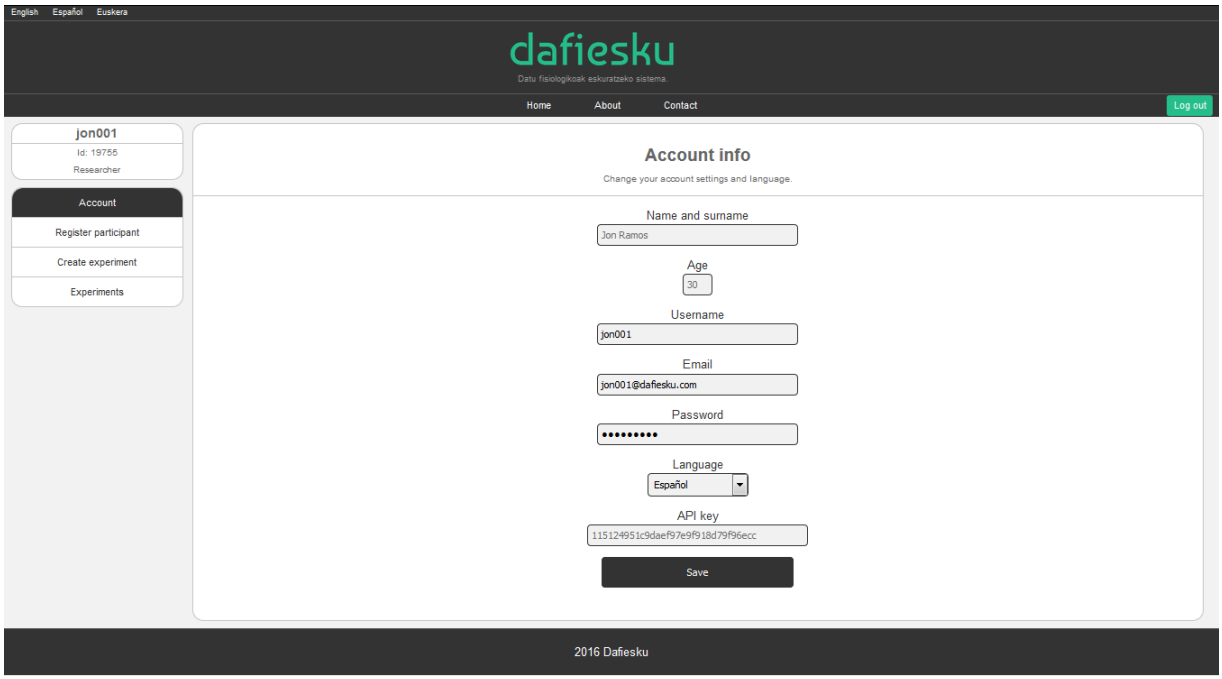


4.7 Irudia: Webgunearen hasierako orria.

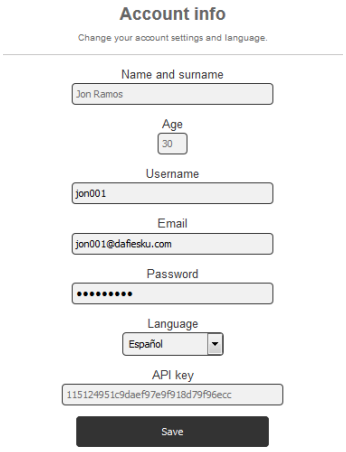
sierako orriak bezala, goiburukoa eta orri-oina estatikoak ditu. Kasu honetan, gainera, goiburukoan logoaren azpian beste menu bat txertatu da. Gehitu diren esteken artean, bakarrik hasierako orrira joateko esteka dago erabilgarri. Beste estekak etorkizunean gehitu nahi badira utzi dira. Menu honen eskubiko aldean saioa amaitzeko botoi bat kokatu da.

Erabiltzaile orriko gorputzeko ezkerreko aldean bi elementu gehitu dira. Alde batetik, erabiltzaile izena, identifikadore zenbakia eta erabiltzaile mota erakutsiko dituen kaxa bat eta, bestetik, hainbat ekintza egin ahal izateko menu bertikal bat. Menu honetan ondorengo aukerak daude:

- Kontua
  Erabiltzaile guztiek haien kontuaren informazioa ikusi eta alda dezakete. Horretarako, ezkerreko menuan “Kontua” botoia sakatu beharko dute (ikus 4.9 irudia). Alda daitezkeen aukerak honakoak dira:
  - Erabiltzaile izena.
  - Email-a.
  - Pasahitza.
  - Hizkuntza.



4.8 Irudia: Webgunearen erabiltzaile orria.

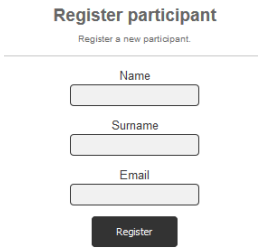


4.9 Irudia: Webgunearen "Kontua" orria.

- Partehartzailea erregistratu

  Ikertzaileek subjektu esperimentalak sisteman erregistra dezakete (ikus 4.10 irudia). Izan ere, esperimentu bat sortzeko orduan subjektua aukeratu behar dute eta hau sisteman erregistratuta egon behar da aurretik.

  Ikertzaile batek partehartzaile bat erregistratzeko honen izena, abizenak eta email-a zehaztu beharko ditu. Partehartzaileak email mezu bat jasoko du esperimentu bat egiteko eskaera duela abisatuz eta sisteman erregistratu beharko da lehen azaldu



4.10 Irudia: Webgunearen "Partehartzailea erregistratu" orria.

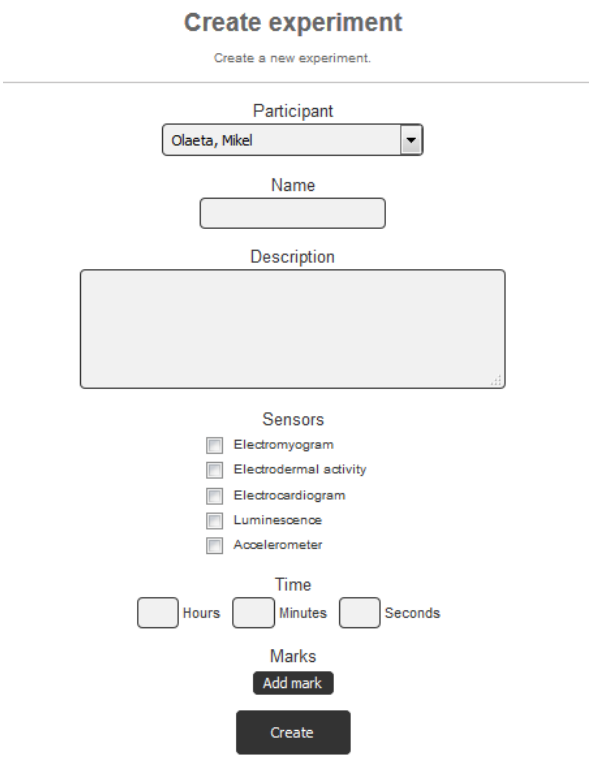
den moduan. Zerbitzaritik email mezuak bidaltzeko *PHPMailer* [45] tresna erabili da, konfiguratzeko nahiko erraza dena. Partehartzailea erregistratzen denean, ikertzaileak ere jasoko du email mezu bat honi buruz jakin dezan, esperimentua sor dezakeela abisatzeko.

- Esperimentua sortu
  Ikertzaileek esperimentu berri bat diseinatu dezakete (ikus 4.11 irudia). Honako informazioa bete beharko dute:

  – Esperimentua burutuko duen partehartzailea edo subjektu esperimentala: sisteman aurretik erregistratuta egon beharko da.
  – Esperimentuaren izena.
  – Esperimentuaren deskribapena.
  – Esperimentuan erabiliko diren sentsoreak.
  – Esperimentuak iraungo duen denbora.
  – Esperimentuak izango dituen ekintzak edo denbora-markak.

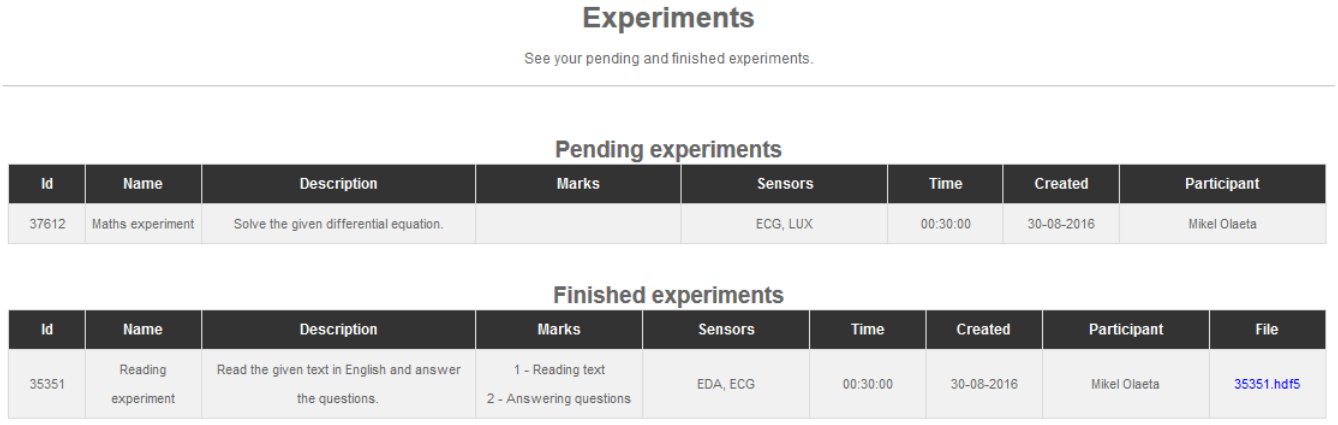
  Esperimentu bat sortzean, hau burutu beharko duen partehartzaileari email mezu bat bidaliko dio zerbitzariak, esperimentu bat esleitu zaiola abisatuz. Ikertzaileak, partehartzaileak esperimentua amaitu arte, itxaron beharko du.

- Esperimentuak
  Erabiltzaile guztiek haien esperimentuen historiala ikus dezakete edozein momentuan (ikus 4.12 irudia). Bertan bi taula agertzen dira. Lehenengo taulan oraindik egiteke dauden esperimentuak eta hauen informazioa ikusiko dira eta, bigarren taulan, amaituta dauden esperimentuak. Azken hauetan, gainera, ikertzaileek esperimentuan jasotako datu fisiologikoek osatzen duten *HDF5* fitxategia deskargatu ahal izango dute, emaitzak aztertu ahal izateko. Partehartzaileek taula berdinak ikusiko



4.11 Irudia: Webgunearen "Esperimentua sortu" orria.

dituzte, hainbat aldaketekin. Alde batetik, ezingo dute *HDF5* fitxategia deskargatu eta, bestetik, esperimentua sortu zuen ikertzailearen izen-abizenak ikusiko dituzte. Ikertzaileek partehartzailearen izen-abizenak ikusiko dituzte zutabe honetan.



4.12 Irudia: Webgunearen "Esperimentuak" orria.

Ezkerreko menuko aukera desberdinen edukia web orriko gorputzean kargatzeko, *jQuery* erabili da. Honi esker, web orri dinamikoak izatea lortzen da eta aukera bakoi-

tzean klik eginez gero, ez da orri osoa (elementu estatikoak barne) kargatu behar. Honen inplementazioan laguntzeko, *js-cookie JavaScript* tresna erabili da.

4.2.3 Biltegiraketa

Atal honetan *HDF5*-ekin inplementatutako programa azalduko da. *API*-aren fitxategiak igotzeko metodoari (*upload*) deia egiterakoan exekutatzen da.

Programaren helburua esperimentua amaitzean jasotako datu fisiologikoak *HDF5* formatuan biltegiratzea da. Horretarako, *Android* aplikazioak nola bidaliko dituen datu hauek zerbitzarira adostu behar izan da Ezekiel Sarasua ikaslearekin. Adostutako prozedura eta formatua ondorengoak izan dira:

- Esperimentuan erabilitako sentsore bakoitzeko testu fitxategi (*.txt*) bat sortuko da. Bertan, sentsore bakoitzarekin jaso diren datuak gordeko dira. Esperimentuan denbora markak definitu badira, marka hauek gordeta dituen fitxategia ere sortuko da. Adibidez:

  ```
  accelerometer.txt
  electrocardiogram.txt
  ...
  marks.txt
  ```

- Esperimentuan erabilitako sentsore guztiak laginketa 100Hz-etara egingo dute. Hau da, sentsoreek segundu batean 100 lagin lortu eta dagokien fitxategian idatziko dituzte. Balioak komaz bereizten dira, azkeneko balioa komaz amaitzen delarik. Seinaleak ondorengo formatuan idatziko dira fitxategietan:

  ```
  X, X, X, ...
  ```

  Maiztasun hori gida gisa erabiliz markak halako portaera jarrai dezakete. Marka bakoitzak zenbaki kode bat izango du, adibidez: *marka1* = 1, *marka2* = 2, ... Beraz, sentsoreen fitxategiekin batera beste fitxategi bat sortuko da antzeko egiturarekin:

  ```
  0, 0, 0, ... 1, 1, 1, ... 0, 0, 0, ... 2, 2, 2, ...
  ```

  0 markarik ez, 1 *marka1*, 0 markarik ez, 2 *marka2*. Beti ere 100Hz-eko maiztasunak errespetatu behar dira, hau da, kasu honetan, segundu bakoitzeko ehun 0, ehun 1 edo

ehun 2 idatzi behar dira. Modu honetan jakin dezakegu esperimentuak irauten duen denboran zehar partehartzaileak zer momentutan sortu dituen markak.

- Fitxategi hauek *Android* aplikazioak sortuko ditu eta *ZIP* batean trinkotuko ditu. Ondoren, *API*-a erabiliz (`POST /api/v1/upload`) zerbitzarira bidaliko da. Metodo honek *HDF5*-ekin inplementatutako *C* programa exekutatuko du, *HDF5* formatuko fitxategia sortuz.

Orain *API*-an inplementatu den *upload* metodoa azalduko da. Metodo honek *Android* aplikazioak jaso diren datu fisiologikoen *.ZIP* fitxategia zerbitzarira bidaltzea ahalbidetzen du.

Beste eskaeretan bezala, *Content-Type* goiburukoa eskuratuko dugu. Fitxategiak irakurtzeko *$_FILES PHP* aldagaia erabiliko dugu. Sistema honetarako fitxategi bakarra igoko den arren, eskaera metodo honek hainbat fitxategi igotzen uzten du. Hortaz, zenbat fitxategi igoko diren kontatuko ditugu. Fitxategiak gordeko diren direktorioa zehaztuko dugu ere (*$path*).

```
<?php
/**
 * Upload files.
 */
$app->post(
'/upload',
function () use ($app, $tm) {
        $contentType = $app->request->headers->get('Content-Type');

        $filesUp = $_FILES;
        $count = count($filesUp);
        $path = $_SERVER['DOCUMENT_ROOT'].'/uploads/';
?>
```

Fitxategi bakoitzeko informazioa gordeko dugu, hau da, izena, errorea egon den ala ez, tamaina, aldi baterako izena eta mota. Ondoren, onartzen diren fitxategi motak gordetzen dituen bektore bat sortuko dugu. Kasu honetan, *.ZIP* fitxategia adierazten duten motak interesatzen zaizkigu.

```
<?php
        if ($count > 0) {
                foreach ($filesUp as $f) {
                        /* Get info from uploaded files. */
                        $fileName = $f['name'];
                        $fileError = $f['error'];
                        $fileSize = $f['size'];
                        $fileTmpName = $f['tmp_name'];
                        $fileType = $f['type'];

                        /* Check files' types. */
                        $acceptedTypes = array(
                        'application/zip',
                        'application/x-zip-compressed',
                        'multipart/x-zip',
                        'application/x-compressed');

                        foreach ($acceptedTypes as $at) {
                                if ($at == $fileType) {
                                        $okay = true;
                                        break;
                                }
                        }
?>
```

Fitxategi bakoitzeko, igotzean errorea ez bada gertatu (*$fileError* == 0), igo den *.ZIP* fitxategiaren kokapenaren *path*-a adierazten duen testu aldagaia sortuko dugu (*$zipPath*). *.ZIP* fitxategia *HDF5* formatura bihurtzeko *exec()* funtzioa erabili da. Honen bidez *C* lengoaian inplementatutako *script*-a exekutatuko dugu. Sortu berri den *HDF5* fitxategiaren kokapena gordeko dugu datu-basean, dagokion ataza eguneratuz.

```
<?php
/* File is uploaded. */
if ($fileError == 0) {
        $newFileName = strtolower($fileName);
```

```
        /* Create string for path to zip file. */
        $zipPath = $path . $newFileName;

        /* Move temp file to given location. */
        move_uploaded_file($fileTmpName, $path.$newFileName);

        /* Execute script to convert the given zip file data to HDF5 file. */
        exec('../../hdf5/to_hdf5_zip '.$zipPath.' 2>&1', $out);

        /* Update task. */
        $id = pathinfo($zipPath, PATHINFO_FILENAME);
        $h5fPath = 'uploads/hdf5/' . $id . '.h5';

        $t = $tm->get($id);
        $t->setHdf5File($h5fPath);
        $result = $tm->update($t);

        if ($result) {
                $response['text'] = 'The file has been uploaded successfully.';
                $response['file'] = $f;
        }
        ...

} else {
        $response['error'] = true;
        $response['text'] = 'Could not upload files.';
}
?>
```

Aurrekoa azalduta, *C* programaren nondik norakoak azalduko dira. Programak sarrera estandarretik (*stdin*) parametro bakarra irakurtzen du, *HDF5* formatura pasa nahi diren datuak dituen *ZIP* fitxategia dena. *ZIP* fitxategia destrinkotu, barruan dauden testu fitxategiak irakurri eta, hauen arabera, datu-multzoak sortuko ditu, *HDF5* formatua duen

fitxategi batean idatziz. *HDF5*-en egitura jarraituz, bi azpitalde sortuko dira *root* taldearen azpian. Bat denbora-markak biltegiratzeko eta bestea sentsoreen datuentzako. Azpitalde bakoitzean beharrezko datu-multzoak sortuko dira, sentsoreen arabera.

Egitura argiago izateko, ikus ondorengo eskema:

```
Group ("/")
|-- Group ("Marks")
        |-- Dataset ("Marks")
|-- Group ("Sensors")
        |-- Dataset ("Electromiogram")
        |-- Dataset ("Accelerometer")
        |-- ...
```

Puntu honetan kodearen zati garrantzitsuenak komentatuko dira. Lehenik eta behin, *HDF5* liburutegia beharko dugu, beste batzuen artean, eta azpitaldeen izenak definituko ditugu:

```
...
#include <hdf5.h>

#define GROUP_NAME1   "/Marks"
#define GROUP_NAME2   "/Sensors"
...
```

Ondoren, fitxategiaren *path*-a eta direktorioaren izena prestatu eta gero, bertan *ZIP* fitxategia destrinkotzen da. Hau egiteko, komando lerroko `unzip` komandoa erabili da.

```
/* Prepare and execute command to unzip file. */
sprintf(f, "unzip %s -d %s", zip_file, zfpath);
x = system(f);

if (x != EXIT_SUCCESS) {
        printf("Error!\n");
        return EXIT_FAILURE;
}
```

Destrinkotutako direktoria irekiko dugu orain, `opendir` funtzioa erabiliz, gero bertako fitxategiak irakurri ahal izateko. Hurrengo pausoa *HDF5* fitxategia eta bi azpitaldeak ("Marks" eta "Sensors") sortzea da. Horretarako, *HDF5* liburutegiak eskaintzen dituen funtzioak erabili behar izan dira. Fitxategia sortzeko, `HDFcreate` funtzioa erabili da, honela definitzen dena:

```
/* Create a new file using default properties. */
file_id = H5Fcreate(dest_file, H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);
```

- Lehenengo parametroak *HDF5* fitxategiaren izena adierazten du.
- Bigarren parametroak fitxategia atzitzeko modua adierazten du:
  - `H5F_ACC_TRUNC`: fitxategia aurretik existitzen bada, edukia ezabatu eta berridatzi egingo da.
  - `H5F_ACC_EXCL`: fitxategia aurretik existitzen bada, errorea emango du irekitzen saiatzean.
- Hirugarren parametroak fitxategia sortzen denerako propietateak adierazten ditu. Fitxategiaren metadatuak kontrolatzeko erabiltzen da, hau da, hainbat egituren tamaina etab. Normalean defektuzko aukera erabiltzen da, `H5P_DEFAULT`.
- Laugarren parametroak fitxategia atzitzen denerako propietateak adierazten ditu. Fitxategiekin egin daitezkeen S/I-ko metodo desberdinak kontrolatzeko erabiltzen da, irakurketa, idazketa eta itxiera, alegia. Kasu honetan ere defektuzko aukera erabiltzen da, `H5P_DEFAULT`.

Orain bi azpitaldeak sortuko ditugu, *root* taldearen azpian. Horretarako `H5Gcreate2` funtzioa erabiliko dugu:

```
/* Create a group named "/Marks" in the file. */
gid1 = H5Gcreate2(file_id, GROUP_NAME1, H5P_DEFAULT, H5P_DEFAULT,
 H5P_DEFAULT);

/* Create a group named "/Sensors" in the file. */
gid2 = H5Gcreate2(file_id, GROUP_NAME2, H5P_DEFAULT, H5P_DEFAULT,
H5P_DEFAULT);
```

- Lehenengo parametroak fitxategi edo talde identifikadorea adierazten du. Kasu honetan, *root* taldearen azpitaldea sortzeko, fitxategi identifikadorea pasako dugu.
- Bigarren parametroak sortuko den taldearen izena adierazten du.
- Hirugarren parametroak taldearen esteka edo lotura sortzen denerako propietate listaren identifikadorea adierazten ditu. Normalean defektuzko aukera erabiltzen da, `H5P_DEFAULT`.
- Laugarren parametroak taldea sortzen denerako propietate listaren identifikadorea adierazten ditu. Kasu honetan ere defektuzko aukera erabiltzen da, `H5P_DEFAULT`.
- Bosgarren parametroak taldea atzitzen denerako propietate listaren identifikadorea adierazten ditu. *API*-aren dokumentazioan dioenez [46], propietate hauek ez daude inplementatuta eta `H5P_DEFAULT` erabiltzea gomendatzen dute.

Hurrengo pausoa hasieran irekitako direktorioan dauden fitxategien edukia irakurtzea da, lehen azaldutako formatua kontuan hartuta.

```
/* Read each file from the directory. */
int buffer[BUF_SIZE], n;
i = 0;
while (fscanf(entry_file, "%d,", &n) > 0) {
        buffer[i] = n;
        i++;
}

int buf_size = i;

/* Create dataspace. */
hsize_t dims[2] = {1, buf_size};
dataspace_id = H5Screate_simple(2, dims, NULL);
```

Buffer batean gordeko ditugu balioak, eta balio kopurua dinamikoa denez, irakurri ahala joango da kontagailu (`i` aldagaia kasu honetan) baten bidez kopuru hau inkrementatzen. Kopuru hau beharrezkoa dugu sentsore bakoitzeko datu-espazioa sortu ahal izateko. Kontagailuko balioa `buf_size` aldagaian kopiatzen dugu eta *HDF5* dagokion datu-espazioa sortuko dugu, `H5Screate_simple` funtzioa erabiliz:

- Lehenengo parametroak datu-espazioaren dimentsio kopurua adierazten du.
- Bigarren parametroak dimentsio bakoitzaren tamaina adierazten du.
- Hirugarren parametroak dimentsio bakoitzaren tamaina maximoa adierazten du. Bigarren parametroaren balioa baino handiagoa izan behar du, baina `H5S_UNLIMITED` (mugagabea) edo NULL (bigarren parametroko berdina) balioak ere zehaztu daitezke.

Sortutako datu-espazioak dimentsio bakarrekoak dira, hau da, bektore bezala biltegiratzea erabaki da. `dims` aldagaian dimentsioak zehazten ditugu, `1xbuf_size` bezala.

Ondoren, fitxategien izenak irakurrita, datu-multzoak sortuko ditugu. Markak dituen fitxategiren bat irakurri bada, datu-multzoa "Markak" azpitaldean sortuko da. Bestela, sentsore bati dagokion fitxategia irakurriko denez, "Sentsoreak" azpitaldean sortuko da.

```
/* Get filenames, removing file formats. */
sscanf(in_file->d_name, "%[^.]", aux);

if (strcmp(aux, "marks") == 0) {
        /* Create the mark dataset. */
        dataset_id = H5Dcreate(gid1, aux, H5T_STD_I32BE, dataspace_id,
        H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
} else {
        /* Create the sensor dataset. */
        dataset_id = H5Dcreate(gid2, aux, H5T_STD_I32BE, dataspace_id,
        H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
}
```

Datu-multzoa sortzeko, `H5Dcreate` funtzioa erabili da.

- Lehenengo parametroak fitxategi edo talde identifikadorea adierazten du. Kasu honetan, dagokion azpitalde identifikadorea pasako dugu.
- Bigarren parametroak sortuko den datu-multzoaren izena adierazten du.
- Hirugarren parametroak datu-motaren identifikadorea adierazten du. Kasu honetan *Integer* motako datuak gordeko ditugu, `H5T_STD_I32BE`.

- Laugarren parametroak datu-espazioaren identifikadorea adierazten du. Lehen sortutako datu-espazioaren identifikadorea pasako dugu.
- Bosgarren parametroak datu-multzoaren esteka edo lotura sortzen denerako propietate listaren identifikadorea adierazten ditu. Defektuzko aukera erabiltzen da, `H5P_DEFAULT`.
- Seigarren parametroak datu-multzoa sortzen denerako propietate listaren identifikadorea adierazten ditu. Kasu honetan ere defektuzko aukera erabiltzen da, `H5P_DEFAULT`.
- Zazpigarren parametroak datu-multzoa atzitzen denerako propietate listaren identifikadorea adierazten ditu. Kasu honetan ere defektuzko aukera erabiltzen da, `H5P_DEFAULT`.

Hurrengo pausoa testu fitxategietatik irakurritako balioak (*buffer*-ean gorde ditugunak) *HDF5* fitxategian idaztea da. Horretarako, `H5Dwrite` funtzioa erabiliko dugu.

```
/* Write the dataset. */
status = H5Dwrite(dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL,
H5P_DEFAULT, buffer);

/* End access to the dataset and release resources used by it. */
H5Dclose(dataset_id);
```

- Lehenengo parametroak idatzi nahi den datu-multzoaren identifikadorea adierazten du. Kasu honetan, dagokion datu-multzoaren identifikadorea pasako dugu.
- Bigarren parametroak memoriaren datu-motaren identifikadorea adierazten du. Kasu honetan *Integer* motako datuak gordeko ditugu, `H5T_NATIVE_INT`.
- Hirugarren parametroak memoriaren datu-espazioaren identifikadorea adierazten du. `H5S_ALL` zehaztuko dugu, datu-espazio osoa aukeratuko dela adierazten baitu, hau da, definitutako dimentsio guztiak idatzi nahi direla.
- Laugarren parametroak datu-multzoaren datu-espazioaren identifikadorea adierazten du. Kasu honetan ere `H5S_ALL` zehaztuko dugu.
- Bosgarren parametroak irakurketa eragiketarako transferentzia propietate listaren identifikadorea adierazten du. Defektuzko aukera erabiltzen da, `H5P_DEFAULT`.

- Seigarren parametroak idatziko diren datuen *buffer*-a adierazten du.

Amaitzeko, sortutako datu-espazioa eta *HDF5* fitxategia itxiko ditugu.

```
/* Terminate access to the dataspaces. */
H5Sclose(dataspace_id);

/* Terminate access to the file. */
H5Fclose(file_id);
```

4.3 Egindako balidazioa

Sistemaren funtzionamendua egokia den frogatzeko balidazio bat egin da maiatzean. Horretarako, esperimentu bat diseinatu da sistemaren bi erabiltzaile motak kontuan hartuta: ikertzailea eta partehartzailea. Egindako probetan, erabiltzaileak ikertzailearen eta subjetu esperimentalaren paperak hartuko ditu, sistemaren bi aldeak frogatuz.

Ikertzailearen papera hartuz, erabiltzaileak esperimentu bat sortuko du, web aplikazioa erabiliz. Subjetu esperimentalaren papera hartuz, erabiltzaile berak lehen sortutako esperimentua burutuko du, sentsoreak eta *Android* aplikazioa erabiliz. Partehartzaileak hainbat ekintza burutzen dituen bitartean sistemak bere datu fisiologikoak jasotzen joango da. Web aplikazioaren eta *Android* aplikazioaren erabilgarritasuna neurtzeko, SUS galdetegiak [47] erabili dira.

4.3.1 Metodoa

Atal honetan balidazioa nola egin den azaltzen da, erabilitako materiala aipatuz, eta amaieran lortutako emaitzen analisi bat eginez.

Partehartzaileak

Bost boluntario (bi emakume) bildu dira, Euskal Herriko Unibertsitateko (UPV/EHU) Informatika Fakultateko ikerketa laborategietatik. Partehartzaileen adina 30-57 urteen artean dago (41,2$\pm$10). Balidazioa egiteko partehartzaileen onespena idatziz jaso da.

Materiala

Balidazioa egiteko Informatika Fakultateko gela bat prestatu da ondorengo gailuekin:

- Web aplikazioa eta web zerbitzua exekutatu dira Ubuntu 14.04 LTS sistema eragilea duen makina birtual batean, eramangarri batean instalatuta. Eramangarri hau erabiliko da ikertzailearen aldea balidatzeko.
- *Android* aplikazioa *Samsung Galaxy Tab* 2.7.0 *tablet*-an (RAM: 1 GB; 1.0 GHz *dual core* prozesadorea; 7" pantaila 1024*600 *pixel*-ekin) instalatu da, *Android* sistema eragilearen bertsioa 4.2.2 izanik.
- *BITalino* [48] sentsore-plataforma edo plaka erabili da partehartzaileen datu fisiologikoak jasotzeko Elektrokardiografo (ECG, *Electrocardiograph*) eta Azalaren jarduera elektrikoa (EDA, *Electro Dermal Activity*) neurtzen duen sentsoreetatik.

Eramangarria eta *tablet*-a *WiFi* bidez konektatu dira, router bat erabiliz. *Tablet*-a eta *BITalino*-a, aldiz, *Bluetooth* bidez konektatu dira.

Bestalde, hainbat dokumentu prestatu dira balidazioa burutzeko:

- Partehartzaileentzat:
  - Balidazioa egiteko onespena.
  - Hainbat datu demografiko jasotzeko galdetegia.
  - Balidazioaren prozedura.
  - Bi SUS galdetegi: bat ikertzaile bezala, bestea subjektu esperimentala bezala.
- Ebaluatzaileentzat:
  - Balidazioaren prozedura.
  - Partehartzaile bakoitzeko formularioa.
  - Ahozko elkarrizketa egiteko galdetegia.

Prozedura

Behin partehartzaileen onespena lortuta, hauen datu demografikoak jaso dira galdeketa baten bidez. Ondoren, balidazioaren nondik norakoak eta egin beharreko atazak azaltzen dituen paper bat eman zaie. Honakoak dira jarraitu beharreko pausoak:

- 1. pausoa: ikertzailea.
  Web aplikazioa erabiliz esperimentu bat definitu behar du partehartzaileak. Emandako paper batean adierazten zaio web aplikazioan sartzeko beharrezko datuak (erabiltzaile izena eta pasahitza) eta esperimentua definitu ahal izateko datuak. Partehartzaileak ziurtatuko du esperimentua ondo sortu dela eta sartutako datuak zuzenak direla.

- 2. pausoa: subjektu esperimentala.
  Partehartzaileak aurreko atazan definitutako esperimentua burutzen du. Horretarako, lehenengo esperimenturako behar diren sentsoreak jantzi behar ditu (paperean azaltzen zaio nola egin), dena ondo funtzionatzen duela egiaztatuz. Ondoren, aurreko pausoan ikertzaile moduan definitutako esperimentua burutu behar du, aipatzen diren atazak eginez.

  Esperimentu honetan partehartzaileak paper batean dagoen testu bat (ingelesez) irakurtzean datza, gero testu horren inguruko galdera batzuk idatziz erantzunez. *Android* aplikazioaren bidez, partehartzaileak adierazi behar du **noiz** ari den testua irakurtzen, galderak irakurtzen eta galderak erantzuten. Behin esperimentua amaitzen duenean, sentsoreak kendu behar ditu.

- 3. pausoa: ikertzailea.
  Ikertzailea bezala berriz ere, partehartzaileak esperimentuan jasotako datu fisiologikoak zerbitzarian ondo jaso direla egiaztatu behar du web aplikazioaren bidez.

Hiru pausoak amaitu eta gero, partehartzaileak bi SUS galdetegiak bete behar ditu. Amaitzeko, partehartzaile guztiei ahozko elkarrizketa bat egin zaie, sistemaren inguruan duten iritzia hobeto ezagutzeko.

Diseinua

Partehartzaileek aurreko atalean azaldutako hiru pausoak jarraitu behar dituzte. Pauso bakoitza burutzeko behar izan duten denbora kalkulatu da. Bi SUS galdetegiak eta ahozko elkarrizketak kontuan hartuta, sistema hobetzeko beharrezko *feedback*-a jaso da.

4.3.2 Emaitzak

Partehartzaile guztiek pauso guztiak burutzea lortu dute. 4.1 taulan pauso bakoitza burutzeko behar izan den denborak azaltzen dira. 2. pausoa (subjektu esperimentala) da burutzeko denbora gehien eskatu duena eta 3. pausoa denbora gutxiena. Prozesu osoa burutzeko behar izan den denbora 25 (*User07*) eta 44 minutu (*User03*) artean egon da.

Erabiltzailea	1. pausoa	2. pausoa	3. pausoa	Guztira
User02	5’	21’	1’	27’
User03	7’	34’	3’	44’
User04	4’	21’	2’	27’
User05	5’	26’	2’	33’
User07	8’	16’	1’	25’
Batezbestekoa	6’	24’	2’	31’
DE *	1,47	6,09	0,75	6,94

* Desbideratze estandarra.

4.1 Taula: Balidazio pauso guztiak burutzeko beharrezko denbora, minituetan.

Sistemaren erabilgarritasunari dagokionez (ikus 4.2 taula), SUS kalifikazioa 78,5$\pm$ 11,25-koa izan da ikertzailearen web aplikazioarentzat eta 71$\pm$15,62-koa *Android* aplikazioarentzat. Kalifikazio hauek 70 baino altuagoak direnez, sistema oso erabilgarria dela kontsidera daiteke [49]. Dena den, bi partehartzaile bi aplikazioak 70 puntu baino gutxiagorekin kalifikatu dituzte, eta beste batek *Android* aplikazioa 42,5 punturekin. Beraz, sistemak hobekuntzak behar dituela nabarmena da.

Erabiltzailea	Ikertzailea	Partehartzailea
User02	77,5	80
User03	65	77,5
User04	67,5	42,5
User05	92,5	87,5
User07	90	67,5
Batezbestekoa	78,5	71
DE *	11,25	15,62

* Desbideratze estandarra.

4.2 Taula: SUS galdetegietan lortutako kalifikazioa.

Egindako elkarrizketetan, partehartzaileek hainbat hobekuntza gomendatu dituzte, batez ere subjektu esperimental bezala esperientzia hobetzeko asmoz. Batek (*User04*) *Android* aplikazioan bertan sentsore desberdinak nola jantzi azaltzen dituzten argibideak

agertzea gomendatu du. Beste partehartzaile batzuek diotenez, 2. pausoa burutzea zaila egin zaie kableengatik. Proiektuan erabili den teknologia dela eta, sentsoreak kabledunak izatea beharrezkoa da. Beraz, ikertzaileak teknologia hau kontuan izan behar du esperimentuak diseinatzeko orduan, emaitzetan eragina izan dezaketen erabilgarritasun arazoak ekiditeko.

4.3.3 Lehen bertsioa

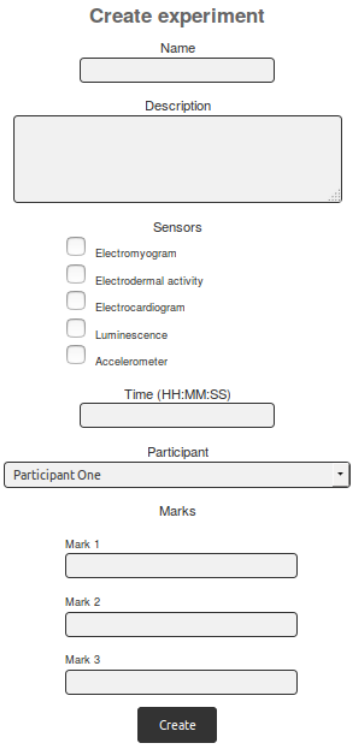
Balidazioa aurrera eramateko inplementatutako web zerbitzuaren eta aplikazioaren hasierako bertsio funtzional bat prestatu da. Bertsio honetan funtzionalitate minimoak landu dira, esperimentazio prozesua arazorik gabe egin ahal izateko. Ondorioz, aplikazioko funtzionalitate batzuk bukatu gabe utzi dira, erregistratze-prozesua, esaterako. Bertsio honetarako ondorengo funtzionalitateak egiaztatu ahal izan dira:

- Erabiltzaile eta web aplikazio arteko elkarrekintza:
  Erabiltzailea bere kontuarekin sisteman sartu eta aplikazioaren erabilera intuitiboa edo argia den.
- Esperimentuen diseinurako prozesua:
  Ikertzaile bezala, esperimentu bat diseinatzearen erraztasuna.
- Web zerbitzuaren funtzionamendu egokia:
  Burututako esperimentuaren egoera eguneratu dela eta *Android* aplikazioaren eta zerbitzariaren arteko komunikazioa ondo egin dela ziurtatu.
- Esperimentuko emaitzaren datuen integritatea:
  Zerbitzarian jaso diren datuak diseinatutako esperimentuarekin bat datozela egiaztatu.

Bestalde, ondorengo funtzionalitate hauek balidaziotik kanpo utzi dira, bai inplementatu gabe edota garapen prozesuan daudelako:

- Erabiltzaileen erregistratze prozesua:
  Probak egitean erabiltzaileak erregistratzerakoan batzuetan arazoak aurkitu dira. Gainera, balidazioan funtzionalitate honen egiaztapenak erabiltzaile bakoitzeko denbora gehiago behar izatea ekartzen du. Izan ere, bi kontu erregistratu beharko lirateke, bat ikertzaile bezala sartzeko eta, bestea, partehartzaile bezala esperimentua burutu ahal izateko.

- Kontuaren konfigurazioa:
  Lehenengo bertsio honetan web aplikazioaren atal hau oraindik ez dagoenez osatuta, ez da bere erabilera egiaztatuko.



4.13 Irudia: Esperimentuaren diseinua egiteko formularioaren lehenengo bertsioa.

Lehen esan bezala, balidazioa egin duten partehartzaileei funtzionalitate hauen inguruan duten iritziari buruz galdetu zaie ahozko elkarrizketan. Esperimentua diseinatzerako orduan, hainbat arazo aurkitu dira.

Balidazioko partehartzaile batzuek komentatu dute sentsoreak aukeratzeko *checkbox*-etan *klik* egin beharra ez dela erosoa. Hortaz, sentsore izenetan *klik* egiteko aukera gehituko da hurrengo bertsiorako.

Gutxienez bostetik bi partehartzailek arazoak izan dituzte formularioko "denbora" eremuarekin. Izan ere, eremu hau testu hutsekoa da eta denbora "HH:MM:SS" formatuan idaztea eskatzen du. Nahiz eta formularioan bertan formatua adierazi (ikus 4.13), bi erabiltzailek gaizki idatzi dute. Ondorioz, bigarren bertsiorako eremu hau erabiltzaileentzako intuitiboagoa egingo da.

Bestalde, erabiltzaileek orokorrean esperimentua diseinatzen bukatzean konturatu dira eremuren bat gaizki idatzi dutela eta, bertsio honetan sistemak ez duenez esperimentuak

editatzen uzten, berri bat sortu behar izan dute. Beraz, hurrengo bertsiorako gutxienez berrespen leiho bat gehituko da, esperimentua sortu baino lehen ikertzaileak sartutako datuak ondo daudela ziurtatzeko.

Aurkitu den beste arazo bat denbora marken "ordena" izan da. Izan ere, esperimentua diseinatzean markak orden finko batean idazten ditu ikertzaileak baina, balidazioak egin ondoren, ikusi da *Android* aplikazioan markak orden desberdinean agertzen direla. Hortaz, konpondu beharreko beste arazo bat litzateke hau. Gainera, bertsio honetaran gehienez hiru marka defini daitezke. Hurrengo bertsioetarako, defini daitezkeen marka kopurua konfiguragarria izango da, eta ikertzaileak erabakiko du zenbat marka behar dituen bere esperimenturako.

Azkenik, ikertzaile bezala esperimentuaren emaitzen fitxategia zerbitzarira igo dela aztertu behar izan dute balidazioaren partehartzaileek. Balidazioa egin aurretik egindako probetan, ikusi da *HDF5* formatura pasatzeko *script*-ak arazoak ematen zituela. Ondorioz, bertsio honetarako esperimentuen emaitzak *ZIP* fitxategi batean igo dira zerbitzarira, *HDF5* formatura bihurketa egin gabe. Gainera, fitxategiak izendatzeko esperimentuari jarritako izena erabili da. Hau etorkizunean aldatu beharko da, esperimentuek izen bera izan baitezakete. Hurrengo bertsiorako fitxategiak *HDF5* formatuan egongo dira atzigarri eta esperimentuaren identifikadore zenbakiarekin izendatuko dira.

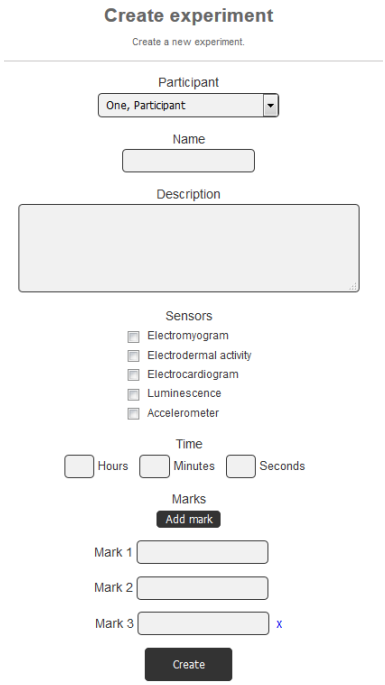
Hortaz, bigarren bertsiorako falta diren funtzionalitateak gehituko dira eta lehen aipatutako arazoak konponduko dira. Esperimentuak ikusteko taula ere argiagoa izateko antolatuko da, bi taula desberdinetan banatuz, esperimentuen egoeraren arabera.

4.3.4 Bigarren bertsioa

Bigarren bertsiorako esperimentazioan aurkitutako hainbat alderdi zuzendu edo hobetu dira, balidazioko partehartzaileen iritziak eta komentarioak kontuan hartuz. Gainera, probak egin eta gero, kodean aurkitutako beste arazo batzuk ere konpondu dira.

Lehenik eta behin, esperimentuak diseinatzeko formularioan aldaketa batzuk egin dira. Alde batetik, partehartzailea aukeratzeko eremua hasierara pasa da (ikus 4.14). Izan ere, ikertzailearentzat erosoagoa da ezer egin baino lehen partehartzailea sisteman erregistratuta dagoen ikustea. Modu honetan, denbora gutxiago galtzen du. Bestalde, denbora eremua aldatu da. Hiru testu-eremutan banatu da, orduak, minutuak eta segunduak idatziz zehazteko.

Beste aldetik, denbora marken eremua dinamikoa egin da. "Marka gehitu" botoian *klik*



4.14 Irudia: Esperimentuaren diseinua egiteko formularioaren bigarren bertsioa.

egitean testu-eremu berri bat gehitzen da. Eremuak ere ezabatu daitezke, eskubi aldean dagoen *x*-n sakatuz.

Formularioko "Sortu" botoian sakatzean, orain berrespen leiho bat agertuko da, sartutako esperimentuaren datuen laburpen batekin, egindako akatsak zuzentzeko aukera izan dezaten ikertzaileek.

Bigarrenik, "Kontua" orriaren inplementazioa egin da, erabiltzaile-kontuaren informazio nagusiena ikusi eta konfiguratu ahal izateko. Adibidez, erabiltzaile-izena, email-a eta pasahitza alda daitezke.

Azkenik, erregistratze-prozesu osoa inplementatu da, hau da, ikertzaileek hasierako orriko erregistro-formularioa erabil dezakete sisteman izena emateko. Behin kontua sortuta, sisteman sartu eta ezkerreko menuko "Partehartzailea erregistratu" aukera sakatuz partehartzaile bat erregistratzeko eskaera egin dezakete formularioa betez.

5. KAPITULUA

Ondorioak eta etorkizunerako lana

Aurkibidea

Azken kapitulu honetan proiektuaren garapenean zehar ondorioztatutako puntu ezberdinak komentatuko dira. Gainera, proiektu honek izan ditzakeen hobekuntzak edo etorkizunean egin daitezkeen gehigarriak azalduko dira.

5.1 Ondorioak

Proiektuaren ikuspuntu orokor bat hartuta, hasieran ezarritako helburu nagusiak bete direla esan daiteke.

Nahiz eta proiektuaren irismenean zehaztuta ez egon, proiektu hau nonahiko konputazioarekin estu erlazionatuta zegoela aipatu beharra dago. Hasiera batean, proiektuaren tutoreekin nonahiko konputazioaren eredua jarraitzen duen sistema bat garatzea adostu zen. Hala ere, ideia hau laster konputazio mugikorragatik ordeztu zen. Izan ere, dedikazio aldetik ezinezkoa litzateke dimentsio horietako sistema GAP batean inplementatzea. Honen inguruko ideiak 5.2 atalan azaldu dira.

Erabilpenari dagokionez, hainbat izan daitezke sistemak eskaintzen dituen aukerak, hala nola:

- Ikerkuntza arloan ikertzaileek subjektu esperimentalen datu fisiologikoak eskuratzeko.
- Medikuntza arloan medikuek pertsonen osasun jarraipena egin ahal izateko.
- Kirol arloan kirolarien datu fisiologikoak edozein ingurunean eskuratzeko.

Bestalde, sistemaren garapena bi ikasleren artean egiteak abantailak eta desabantailak dauzka. Alde batetik, proiektuaren hainbat ataletan elkar lagundu gara. Bestetik, ordutegiak eta lan erritmo bera jarraitu behar izan beharra arazoak ekar ditzake. Dena den, taldeko lana oro har ondo kudeatu dugula uste dugu, ezustekoak egon diren arren.

Esperientzia ona izan da guretzat landutako sistemari buruz artikulu bat UCAmI'2016 kongresura bidaltzeko aukera eduki izana eta bertan onartua izatea (ikus memoria honetako A eranskina). Gainera, artikulurako egindako sistemaren balidazioak garapena hobetzeko lagundu gaitu, aplikazioen erabilgarritasuna neurtuz.

Proiektu honetan guretzat garapen aldetik lehendik ezezagunak ziren hainbat teknologia berriak ikusi ahal izan ditugu. Aipatu beharra dago teknologia hauen erabilpena gaur egungo informatika munduan oso erabiliak direla.

Garapenean zehar izandako arazoak hainbat izan dira, batez ere erabili diren teknologien ezjakintasunagatik denbora asko pasa da prestakuntza atala lantzen eta ondoren inplementazio fasearekin, aurkitutako akatsak eta arazoak konpontzen. Dena den, ikasitako lezioak asko izan dira eta etorkizunean proiektu honekin ikasitakoarekin erlazionatutako lanaren bat egiteko prozesua arinagoa izango dela espero da.

Azkenik, eskertzekoa izan da zuzendari eta zuzendarikideaz gain, Borja Gamecho doktorearen laguntza eduki ahal izana. Aditu gisa, hainbat kontuetan lagundu gaitu, batez ere *Android* aplikazioarekin, txartelen kudeaketarekin eta sistemaren balidazioarekin.

5.2 Etorkizunerako lana

Gradu Amaierako Proiektua amaitu den arren, honi jarraipena eman diezaioketen lan dezente egin daitezke, batez ere, sistemaren portaera findu eta erabilpena zabaltzeko. Hona hemen proposatutako hobekuntza posible batzuk:

- 5.1 atalan aipatu den moduan, proiektu hau nonahiko konputazioaren ideia kontuan hartuta diseinatu zen hasiera batean. Honako helburuak lortu nahi ziren:
  - Datu fisiologikoak eskuratzeko txartelaren bat detektatuz gero automatikoki konektatzea.
  - Edozein datu fisiologikoen txartela izanda datu fisiologikoak eskuratzeko aukera izatea.
  - Gailu desberdinak erabiliz erabiltzaileek sisteman egin ditzaketen ekintzak berdinak izatea, adibidez, partehartzaileek bai mugikorra nola konputagailutik esperimentuak burutzeko aukera izatea.
  - Laino sistema baten erabilpena: datu fisiologikoak biltegiratu eta edozein lekutik atzitzeko aukera izatea.

  Etorkizunerako lan bezala, interesgarria izan daiteke ezaugarri hauen inplementazioa egitea, sistemaren erabilpen aukerak zabalduz.

- Web zerbitzua edo *API*-a inplementatzeko, beste *framework* baten erabilera interesgarria izan daiteke, konplexutasuna gehitzen duen arren, aukera gehiago eskaini ditzake.

- Web aplikazioaren interfazeak hobekuntzak izan ditzake, bai erabilera aldetik nola irismenean. Gainera, nabigatzaile desberdinekin bateragarritasuna gehitzea ideia ona izango litzateke.

- Web aplikazioa dinamikoagoa eta intuitiboagoa izateko aldaketak egitea komeni da, batez ere erabiltzaile esperientzia hobetzeko. Hau lortzeko, aplikazioak eskaintzen dituen aukerak gehitu beharko lirateke, egiteke utzi den *feedback* sistema, esaterako. Hobekuntza hauei esker, ikertzaile-partehartzaile elkarrekintza asko hobetuko litzatekela uste da.

- Azkenik, kodearen egitura hobe daitekeela uste da, batez ere eskalagarritasuna suspertzeko eta sistemari funtzionaltasunak gehitzeko prozesua ahalik eta arinena izateko.

Eranskinak

A. ERANSKINA

UCAmI'2016 kongresurako idatzitako artikulua

Oharra: argitaratuko dena aldaketatxo batzuk izan ditzake memoria honetan eransten den bertsioarekiko.

Physiological Data Acquisition System Based on Mobile Computing

Ezekiel Sarasua, Maider Simón, Borja Gamecho, Edurne Larraza-Mendiluze and Nestor Garay-Vitoria

Egokituz Laboratory, Informatika Fakultatea (UPV/EHU); Manuel Lardizabal pasealekua 1; E-20018 Donostia (Gipuzkoa)
nestor.garay@ehu.eus

Abstract—The way to achieve enough data for data mining is accessing existing databases or directly acquiring data with the aim of creating new databases. In this paper we present the DAFIESKU system built to acquire different types of physiological data via experiments and the factors taken into account when developing it, in order to facilitate the creation of new datasets by means of mobile and wearable devices. DAFIESKU has been evaluated on a case study associated to non-classroom learning.

Keywords: Physiological data acquisition; Data storage; Mobile and Wearable Computing; Design of experiments.

1 Introduction

Before analyzing or applying machine learning techniques to physiological signals, datasets have to be obtained. There are a huge number of public repositories storing datasets which are available online. Nevertheless, in some cases researchers need to create new datasets to work with, being the task of acquiring the datasets an essential step in the research process.

The way data are obtained is an important issue related to the validity of considered data. The main approaches to record data are in controlled environments such as research laboratories or in uncontrolled environments such as in real life.

In this paper authors review topics to be taken into account and present a system aimed at acquiring physiological data on a mobile uncontrolled environment named DAFIESKU. This system allows researchers to conduct experiments remotely with the collaboration of participants, taken ethical and privacy issues into account.

2 Related Work

It is not surprising to find many frameworks, toolkits and platforms devoted to the remote evaluation in the scientific literature [1, 2]. Last decade, the high popularity of smart phones and its applications, commonly called apps, presented an opportunity to

make remote evaluations in mobile environments (for instance, Funf [3] and Aware [4]). In recent years, the use of physiological signals has emerged in the computer science area due to the vast amount of wireless wearable devices that provide such signals. Due to this fact, we can find in the literature similar systems to record and store physiological signals such as Biosignal Ignitor toolkit [5] and Physiodroid [6].

The main objective of DAFIESKU is to provide resources and tools to researchers in order to facilitate the deployment of an experiment from its design to the analysis of the results, which are not usually covered in the references found in literature.

3 Design issues

Design issues on the development of DAFIESKU system are here explained.

3.1 Mobile Sensors, Data Acquisition and Transmission

Nowadays, sensors are being more comfortable due to miniaturization. Normally, these sensors are integrated in all-in-one hardware platforms, such as Biosignalsplux [7] or Shimmer [8]. Nevertheless, these platforms still have limited storage capabilities and personal area network connectivity.

In order to overcome these limitations, a usual solution is to combine sensor platforms with mobile phones carried by users. Thanks to mobile devices, data obtained from sensors are transmitted first to the storage of the mobile device and finally to a system server available on the Internet. The main idea of DAFIESKU system is to make possible the acquisition of data in real life, without limiting it to controlled environments. These data will better reflect user natural characteristics and behavior.

3.2 Ethical Data and Privacy for Storage

Ethical and privacy issues have also to be taken into account. Physiological data are in most of the cases personal and have to be securely stored in order to ensure the people behind these data are not identified.

3.3 Further Data Analysis

Once all the ethical and privacy issues considered, data should be processed and filtered, and in order to make further analysis easier, standards, such as HDF5 [9] followed.

4 DAFIESKU System

DAFIESKU system has been designed to take advantage of wireless devices in order to obtain the less intrusive environment for the participants in the data acquisition. DAFIESKU consists of a set of wearable devices with physiological sensors coordi-

nated by a server. Wearable devices with wireless connections usually offer Bluetooth based connectivity which could be used to send physiological data to any computer. For this reason, a mobile device (e.g. smart phone or tablet) is used as intermediary between the server and the wearable devices (see Fig. 1). The main advantage of this approach is the possibility it gives to configure the data acquisition in each experiment and to guide and support the participants during the experiments.

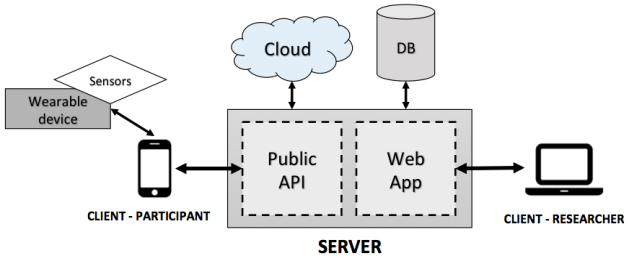


Fig. 1. Architecture of DAFIESKU system.

4.1 Configuring Data Acquisition

With DAFIESKU, researchers may establish which data, how and for how long will be acquired for creating databases with physiological information. The data to be gathered depends on the scope of the studies that the researchers plan to develop in the future, the sensors that are available for acquisition, the population, etc.

DAFIESKU system gives the researchers a web client, which offers the possibility to define how sensors (among the available ones) will be used, whether they will be used together or separately, and which kind of information will be recorded.

4.2 Providing guidance to the Participants

In DAFIESKU system, participants play a key role in the data acquisition. Among others, they decide when to start the experiments and to provide feedback to label the physiological data. In order to facilitate these activities, the researcher must also: 1) specify the experiment description and define the tasks for the participants, 2) describe the instructions to attach the sensors, and 3) set a list of predefined marks or activities related to the experiment to label the data set. All these texts must be comprehensive for the participants in order to avoid problems during the experimentation.

4.3 Experiment development

The participants will be able to download the experiment to their Android mobile device (smart phone or tablet). Once there, they will have to set the sensors and connect them to the mobile device via Bluetooth. Finally they will have to start recording data and labeling moments, following the instructions of each experiment. These last two will be mere “push button” or “select from list” tasks, respectively.

Tasks are classified as being pending or finished, depending on their status. In order to perform a task, it has to be selected from the pending tasks list. Clicking on a finished task name you can get the task feedback.

Once the user has selected a task from pending list, a screen shows all the information regarding to the selected task (description, sensors participant has to wear, maximum time estimation, and activities to perform). In order to make correct recordings, the user can check whether sensors are correctly adjusted before performing activities related to the tasks.

5 Case study to evaluate DAFIESKU system on non-classroom learning experimentation

A case study was carried out to test the adequacy of the described system. For this purpose a representative experiment on non-classroom learning was designed for DAFIESKU in order to test both roles in the system. As a researcher, the experimental subject introduced the information to create the experiment. On the other hand, as a participant of experiments, the same experimental subject performed fixed tasks while his/her physiological data were acquired. The usability of both web and mobile applications was measured by using SUS questionnaires [10].

5.1 Method

Participants and Apparatus

5 volunteers (2 females) were recruited from the surrounding research laboratories of the Faculty of Informatics of the University of the Basque Country (UPV/EHU). The experimental subjects ranged from 30 to 57 years old (41.2±10). Informed consent was obtained from all individual experimental subjects included in the study.

The Web service for designing experiments run on a virtual machine with Ubuntu 14.04 LTS operating system. This Web service was used when playing researcher role in this experiment using a laptop. Designed Apps for acquiring data run on a Samsung Galaxy Tab 2.7.0 with Android 4.2.2 operating system (RAM: 1 GB; 1.0 GHz dual core processor; 7" screen with 1024*600 pixels). BITalino [11] sensor platform was used to collect physiological data from the Electrocardiography (ECG) and Electro Dermal Activity (EDA) sensors. The virtual machine running on the laptop and the tablet were connected wirelessly using a WiFi router. The tablet and the BITalino were connected via Bluetooth interface.

Procedure

After consent has been obtained from experimental subjects, their demographic data were gathered. Then, a paper with the tasks to complete was delivered to each experimental subject.

Task 1 was defining an experiment with certain characteristics by using the Web client of DAFIESKU. This was made with the researcher role. Task 2 was made as

participant. They prepared the equipment needed in the experiment, verified they worked correctly and then they made the activities needed on the experiment defined on Task 1. This experiment consisted on reading a text printed on paper, and then reading and answering several questions related to that text, also on a paper and using a pen, like they were making exercises out of classroom. Participants had to indicate by using the system when they were reading the text, when reading questions and when answering questions. After the experiment finished, they had to put out sensors. As a researcher again, on Task 3 they verified by using the Web client again that the data of the experiment had been stored on the server.

When finishing these three tasks, experimental subjects completed two SUS questionnaires, one as researcher and the other as participant of physiological experiment. Finally, they were interviewed by up to two DAFIESKU design team members in order to get more feedback.

Design

Experimental subjects had to follow a routine composed of the abovementioned three tasks. Time required to complete each task were measured. The experimental subjects provided qualitative feedback by means of two SUS questionnaires, one for the researcher role and the other for the participant role, and the final interviews.

5.2 Results and Discussion

All the experimental subjects successfully completed tasks required. Task 2 was the one needing more time while Task 3 was the one needing less time. Minutes needed for completing three tasks went from 25' (User07) to 44' (User03). Concerning the usability of the system, the SUS scores were 78,5±11,25 for the researchers Web client application and 71±15,62 for the Android application. These scores are over 70 and therefore both applications may be considered acceptable from the usability point of view [10].

Experimental subjects also suggested several enhancements, almost with the aim of making a better participant experience for data acquisition. As DAFIESKU aims to create experiments and register data, including in the APP help or instructions to correctly adjust sensors is a good idea for future versions.

6 Conclusions and Future Work

In this paper we have presented DAFIESKU system, developed to configure data acquisition, do the collection and finally store data. Its initial aim is preparing experiments aimed at being made by students to analyze their physiological characteristics while non-classroom learning, but it could be generalized to other experimental topics and settings.

A case study has been carried out to test the validity and usability of the described system. Participants were doing fixed tasks such as reading texts and questions, and

answering questions. Achieved results are promising and show several enhancements that are being currently made on DAFIESKU system.

Next, a new version of DAFIESKU will be deployed to define experiments and acquire physiological data. Once the acquisition has been finished, data will be filtered and processed, considering all the ethical, legislation security and validity issues, in order to make data available for further research.

Acknowledgements

This work has been supported by the Ministry of Economy and Competitiveness of the Spanish Government and by the European Regional Development Fund (projects TIN2013-41123-P and TIN2014-52665-C2-1-R), and by the Department of Education, Universities and Research of the Basque Government under grant IT395-10. Last three authors belong to the Basque Advanced Informatics Laboratory (BAILab), grant UFI11/45, supported by the University of the Basque Country (UPV/EHU). B. Gamecho is supported by "Convocatoria de contratación de doctores recientes hasta su integración en programas de formación postdoctoral en la UPV/EHU 2015".

References

1. Paternò, F. (2003). Tools for remote web usability evaluation. In HCI International (pp. 828-832).
2. Christos, F., Christos, K., Eleftherios, P., Nikolaos, T., & Nikolaos, A. (2007). Remote usability evaluation methods and tools: A survey. In Proceedings of the 11th Panhellenic Conference in Informatics (PCI 2007) (pp. 151-162).
3. Aharony, N., Pan, W., Ip, C., Khayal, I., & Pentland, A.. Social fMRI: Investigating and shaping social mechanisms in the real world. Pervasive and Mobile Computing, 7(6), 643-659. (2011)
4. Ferreira, D.: Aware: A mobile context instrumentation middleware to collaboratively understand human behavior. In: Ph.D. dissertation, University of Oulu, Fac- ulty of Technology. (2013)
5. Silva, H. P., Lourenço, A., Fred, A., & Martins, R. (2014). BIT: Biosignal igniter toolkit. Computer methods and programs in biomedicine, 115(1), 20-32.
6. Banos, O., Villalonga, C., Damas, M., Gloesekoetter, P., Pomares, H., & Rojas, I. (2014). Physiodroid: Combining wearable health sensors and mobile devices for a ubiquitous, continuous, and personal monitoring. The Scientific World Journal, 2014.
7. Biosignalsplux. http://biosignalsplux.com/index.php/en/
8. Shimmer. http://www.shimmersensing.com/
9. M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, An overview of the HDF5 technology suite and its applications. In Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, ACM, 2011, pp. 36-47.
10. Brooke, J. (2013). SUS: a retrospective. Journal of usability studies, 8(2), 29-40.
11. H. Silva, A. Fred, and R. Martins. Biosignals for Everyone. Pervasive Computing, IEEE 13(4), 2014, pp. 64-71.

B. ERANSKINA

Erabilpen gida

Aurkibidea

B.1 Sarrera

Eskuliburu honen bidez erabiltzaileei DAFIESKU sistemaren funtzionamenduaren azalpen labur bat eta erabilpen gida lagungarri bat eskaintzea da. Sistema erabiltzeko pauso edo ekintza desberdinak azalduko dira.

B.2 Sisteman izena eman

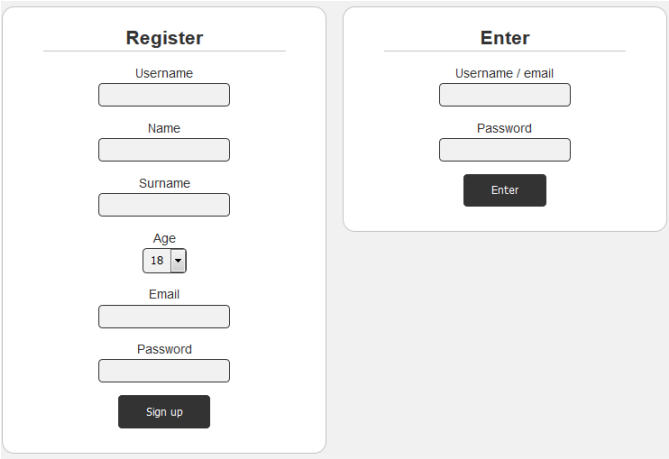
DAFIESKU sistema erabiltzeko, izena eman beharra dago sisteman. Bi erabiltzaile mota desberdintzen dira, alde batetik ikertzaileak eta, bestetik, partehartzaileak edo subjektu esperimentalak.

Ikertzaileek sisteman erregistroa egin dezakete zuzenean, subjektu esperimentalek ez bezala. DAFIESKU sistemaren ideia ikertzaileentzat esperimentu desberdinak diseinatzeko tresna bat eskaintzea da.

Izena emateko, DAFIESKU web aplikazioa (`https://dafiesku.com`) nabigatzailean ireki eta hasierako orrian agertzen den erregistro formularioa (ikus B.1 irudia) bete beharko du ikertzaileak. Eskatzen diren datuak ondorengoak dira:

- Erabiltzaile izena: ez errepikagarria.
- Izena.
- Abizenak.
- Adina.
- Email-a: ez errepikagarria.
- Pasahitza.

Behin formularioa bete dela, hurrengo pausoa kontua aktibatzea da. Horretarako, automatikoki zerbitzariak email bat bidaliko du eta email-an egongo den *URL* baten bidez aktibatuko da kontua eta ikertzailea sisteman sartzeko aukera izango du.



B.1 Irudia: Webgunearen erregistro eta sisteman sartzeko formularioak.

B.3 Sisteman sartu

Sisteman sartzeko, DAFIESKU web aplikazioa (`https://dafiesku.com`) nabigatzailean ireki eta hasierako orrian agertzen den sartzeko formularioa (ikus B.1 irudia) bete beharko du erabiltzaileak. Kontuan izan behar da sisteman sartzeko kontua sortuta eta aktibatuta egon behar dela.

B.4 Sistematik atera

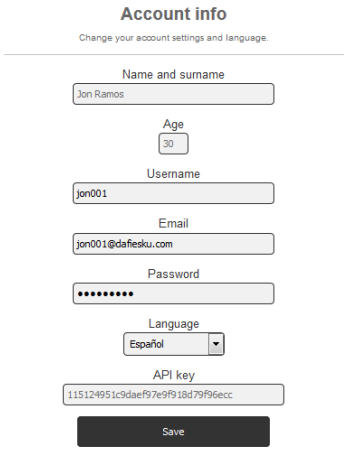
Sistematik ateratzeko edo saioa amaitzeko, goiburuko eskubiko atalean dagoen “Saioa amaitu” botoia sakatu.

B.5 Kontuaren informazioa ikusi eta aldatu

Erabiltzaile guztiek haien kontuaren informazioa ikusi eta alda dezakete. Horretarako, ezkerreko menuan “Kontua” botoia sakatu beharko dute. Alda daitezkeen aukerak honakoak dira (ikus B.2 irudia):

- Erabiltzaile izena.
- Email-a.
- Pasahitza.

- Hizkuntza.



B.2 Irudia: Webgunearen "Kontua" orria.

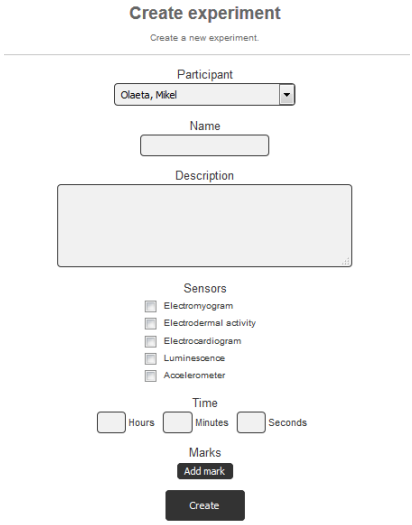
Aldaketak gordetzeko, "Gorde" botoian sakatu.

B.6 Esperimentuak sortu

Ikertzaileek ezkerreko menuko "Esperimentua sortu" botoia sakatuz, esperimentu berri bat diseinatu dezakete. Honako informazioa bete beharko dute (ikus B.3 irudia):

- Esperimentua burutuko duen partehartzailea edo subjektu esperimentala: sisteman aurretik erregistratuta egon beharko da. Ez bada horrela, ikusi B.7 atala.
- Esperimentuaren izena.
- Esperimentuaren deskribapena.
- Esperimentuan erabiliko diren sentsoreak.
- Esperimentuak iraungo duen denbora.
- Esperimentuak izango dituen ekintzak edo denbora-markak.

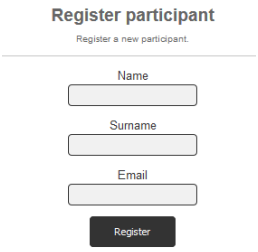
Esperimentua sortzeko, "Sortu" botoian sakatu. Esperimentu bat sortzean, hau burutu beharko duen partehartzaileari email mezu bat bidaliko dio zerbitzariak, esperimentu bat esleitu zaiola abisatuz. Ikertzaileak, partehartzaileak esperimentua amaitu arte, itxaron beharko du.



B.3 Irudia: Webgunearen "Esperimentua sortu" orria.

B.7 Partehartzaile bat erregistratu

Ikertzaileek subjektu esperimentalak sisteman erregistra dezakete. Izan ere, esperimentu bat sortzeko orduan subjektua aukeratu behar dute eta hau sisteman erregistratuta egon behar da aurretik. Hortaz, partehartzailea erregistratzeko, ezkerreko menuko "Partehartzailea erregistratu" botoia sakatu beharko du ikertzaileak (ikus B.4 irudia).

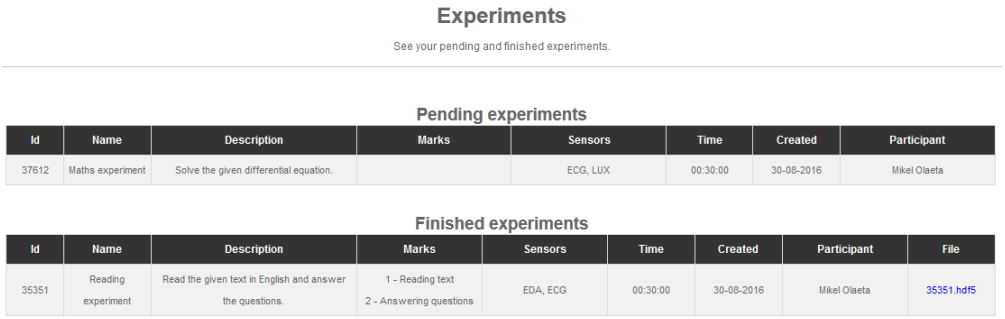


B.4 Irudia: Webgunearen "Partehartzailea erregistratu" orria.

B.8 Egindako eta egiteke dauden esperimentuen informazioa ikusi

Erabiltzaile guztiek haien esperimentuen historiala ikus dezakete edozein momentuan. Bertan bi taula agertzen dira (ikus B.5 irudia). Alde batetik, oraindik egiteke dauden es-

perimentuak eta hauen informazioa eta, bestetik, amaituta dauden esperimentuak. Azken hauetan, gainera, ikertzaileek esperimentuan jasotako datu fisiologikoek osatzen duten *HDF5* fitxategia deskargatu ahal izango dute, emaitzak aztertu ahal izateko.



Experiments

See your pending and finished experiments.

Pending experiments

Id	Name	Description	Marks	Sensors	Time	Created	Participant
37612	Maths experiment	Solve the given differential equation.		ECG, LUX	00:30:00	30-08-2016	Mikel Olaeta

Finished experiments

Id	Name	Description	Marks	Sensors	Time	Created	Participant	File
35351	Reading experiment	Read the given text in English and answer the questions.	1 - Reading text 2 - Answering questions	EDA, ECG	00:30:00	30-08-2016	Mikel Olaeta	35351.hdf5

B.5 Irudia: Webgunearen “Esperimentuak” orria.

C. ERANSKINA

Makina birtuala sortu

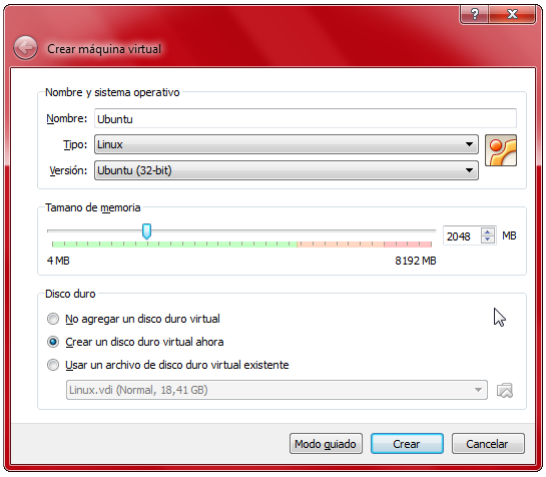
VirtualBox erabiliko dugu makina birtuala sortzeko. Erabilitako bertsioa 5.0.14 izan da, baina edozein bertsioarekin jarraitu beharreko pausoak berdinak dira.

Egin beharreko lehenengo pausoa sortu nahi dugun makina birtualaren sistema eragilea aukeratu eta honen irudia (*.iso* formatuan normalean) deskargatzea izango da. Kasu honetan *Linux Ubuntu 14.04* bertsioa instalatuko dugu, 32 bitekoa. Behin irudia deskargatu dugula, *VirtualBox* ireki eta "Nueva" botoia sakatuko dugu.

C.1 Irudia: Makina birtuala sortzen - 1.

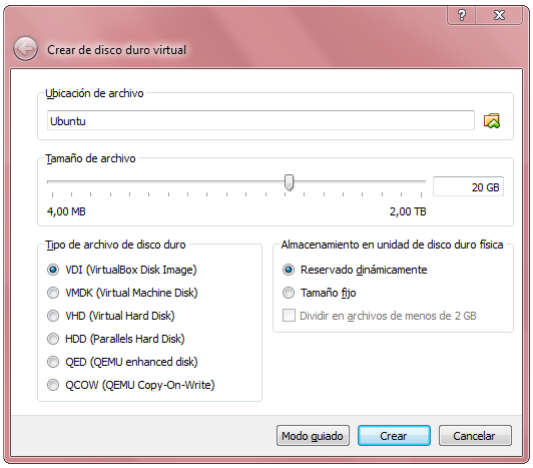
Agertuko den pantailan (C.2 irudia) makina birtualaren izena eta instalatuko dugun sistema eragilea aukeratuko ditugu. Makina birtualak izango duen memoria tamaina ere esleitu beharko dugu. Kasu honetan 2 GB esleitu dira, erabilitako ordenagailuak 8 GB baititu, eta modu erosoan lan egiteko beharrezkoak direla pentsatu da. Ondoren, disko gogor birtual berria sortzeko esango diogu. "Crear" botoia sakatuko dugu hurrengo pausora joateko.

Disko gogor birtualari dagokionez, fitxategiaren kokapena dagoen moduan utziko dugu (C.3 irudia). Kasu honetan 20 GB emango dizkiogu, oso azkar bete ez zezan. Defektuz



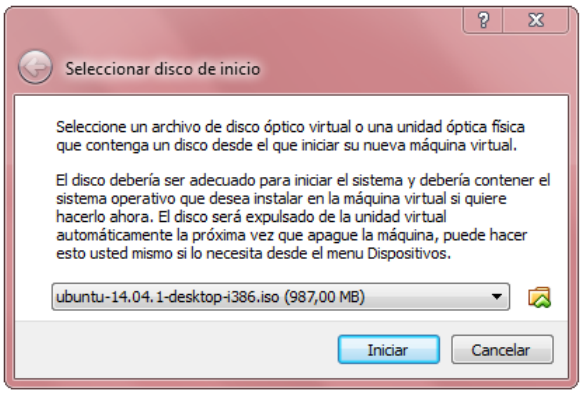
C.2 Irudia: Makina birtuala sortzen - 2.

aukeratuta dagoen mota (VDI) utziko dugu eta disko gogorraren unitate fisikoaren biltegiratze mota dinamikoa aukeratuko dugu. Modu honetan, espazio gabe geratzen bada, dinamikoki joango da tamaina gehiago ematen.



C.3 Irudia: Makina birtuala sortzen - 3.

Behin hau amaituta, “Iniciar” botoia sakatuko dugu sistema eragilea instalatzeko (C.4 irudia). Sistemaren irudia aukeratu eta “Iniciar” sakatuko dugu. Instalazio pausoak jarraituko ditugu eta makina birtuala prest izango dugu erabiltzen hasteko.



C.4 Irudia: Makina birtuala sortzen - 4.

D. ERANSKINA

Bilera aktak

Proiektuaren garapenaren zehar egindako bilera guztien nondik norakoak bildu dira eranskin honetan.

Aurkibidea

Bilera guztiak leku berean eta partehartzaile berdinekin egin dira:

Lekua: Informatika Fakultateko 2.1 mintegia

Bilerara hurbilduak: Ezekiel Sarasua, Maider Simón, Nestor Garay eta Edurne Larraza

D.1 Konstituzio bilera

Data: 2015/06/05, 13:00

Helburuak

Proiektuaren nondik norakoak komentatu.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Proiektuaren irismena finkatu: lana bi ikasleen artean banatu eta amankomuna finkatu.

D.2 Lehenengo bilera

Data: 2015/09/04, 15:00

Helburuak

Proiektuaren irismenaren berrikuspena.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Proiektuaren Helburuaren Dokumentua landu.

D.3 Bigarren bilera

Data: 2015/09/17, 15:00

Helburuak

Proiektuaren Helburuaren Dokumentuaren berrikuspena. Zuzendariei proiektuan lan egiteko toki bat eskatu.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Proiektuaren Helburuaren Dokumentua landu.
- Nestorrek eta Edurnek:
  Libre dagoen laborategiren bat bilatu.

D.4 Hirugarren bilera

Data: 2016/02/01, 10:30

Helburuak
Prestakuntza fasearen berri eman eta azken berrikuspena egin.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Garapenarekin hasi eta Proiektuaren Helburuen Dokumentuaren azken zuzenketak egin.

D.5 Laugarren bilera

Data: 2016/02/22, 10:45

Helburuak
Hasierako demo bat erakutsi, Proiektuaren Helburuen Dokumentuaren berrikuspena komentatu eta proiektuan eman beharreko hurrengo pausoak argitu.

Egin beharrekoak

- Ezekielek:
  BITalino eta *Android* arteko elkarrekintzarekin lan egiten jarraitu eta sentsoreetatik jasotako datuak tratatu eta modu argiago batean pantailaratu.
- Maiderrek:
  Erabiltzaileak sisteman erregistratzeko atalan lan egin eta funtzionalitate berriak gehitu.

D.6 Bosgarren bilera

Data: 2016/04/04, 10:45

Helburuak

Zalantzak argitu. Bigarren demo bat erakutsi. Orain arte egindakoa azaldu. Egiteke geratzen dena antolatu. Proiektuan eman beharreko hurrengo pausoak argitu. Proiektua ICNR2016 [50] kongresuan aurkeztearen inguruan hitz egin.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Jarraipen eta kontrola. Plangintzaren berrikuspena: atazak birdefinitu. Atazak (esperimentuak) sortzeko sistemaren garapena egin. HDF5 formatuaren inguruan informazioa bilatu. Cloud zerbitzuen inguruan informazioa bilatu.
- Ezekielek:
  Android aplikazioaren garapenarekin jarraitu.
- Maiderrek:
  Datu-basea hobetu antolatu eta osatu. Kodea txukundu. API-a osatu eta zuzendu. Web aplikazioari itxura eman. Aplikaziorako proba kasuak definitu.
- Denok:
  Kongresurako artikulua prestatu.

D.7 Seigarren bilera

Data: 2016/05/02, 10:45

Helburuak

Proiektuaren egoeraren berrikuspena. Kongresura bidalitako artikuluaren egoeraren eguneraketa.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Proiektuaren garapenarekin jarraitu eta aurkitutako akatsak zuzendu.

D.8 Zazpigarren bilera

Data: 2016/05/30, 10:30

Helburuak

ICNR2016 kongresura bidalitako artikuluaren eguneraketa. Proiektua UCAmI [1] kongresuan aurkeztearen inguruan hitz egin. Web aplikazioaren interfazearen diseinua komentatu.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Sistemaren balidazioa egin ahal izateko bertsio egonkor bat landu.
- Maiderrek:
  Esperimentuak definitzeko interfazea landu.
- Nestorrek eta Edurnek: Esperimentua diseinatu eta prestatu.

D.9 Zortzigarren bilera

Data: 2016/06/13, 10:30

Bilerara hurbilduak: Borja Gamecho

Helburuak

Egin beharreko sistemaren balidazioa prestatu. Kongresuko artikulua osatu.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Kodean aurkitutako akatsak zuzendu. Aplikazioaren pantailazoak egin eta hauen deskribapena idatzi.
- Maiderrek:
  Balidazioa egiteko 10 ikertzaile eta 10 partehartzaile kontu sortu.

D.10 Bederatzigarren bilera (1. balidazioa)

Data: 2016/06/14, 9:30

Bilerara hurbilduak: Borja Gamecho

Helburuak

Sistemaren balidazioa egin bost partehartzaileekin. Kongresuko artikulua osatzen bukatu.

Egin beharrekoak

- Ezekielek eta Maiderrek:
  Partehartzaileek aipatutako komentario guztiak kontuan hartuta sisteman aldaketak egin. Aurkitutako akatsak eta arazoak konpondu. Artikuluaren azken berrikuspena.
- Maiderrek:
  Web aplikazioaren interfazea txukundu. Esperimentuen atala osatu: Denbora marka kopurua dinamikoa izatea. Datuen konfirmazioa. "Denbora" atala berregin. Denbora markei "ordena" gehitu. Hizkuntza aldaketa egitean aurkitu diren arazoak konpondu. *API*-a zuzendu. *HDF5* formatuko C fitxategia konpondu.

D.11 Hamargarren bilera

Data: 2016/07/18, 15:00

Helburuak

GAP-en memorien egoera eta aldaketak egiteko plangintza. Zuzendarien oniritziak: noizko eta zein baldintzetan. GAUR-en bidez egin beharreko pausuak (aurreko puntuarekin oso lotuta). Entsaioak defentsak egin aurretik. UCAmI2016 artikulua. Bestelakoak.

Egin beharrekoak

GAP-en laburpena eleanitza. Eranskinak ordenatu. GAUR-en sartu eta defentsa eskaera egin. UCAmI2016 artikuluaren berrikuspena eta zuzenketak.

D.12 Itxiera bilera

Data: 2016/08/29, 11:00

Helburuak

Memoriaren azken bertsioa komentatu. Artikuluaren egoera komentatu eta aldaketak adostu. Defentsa prestatzeko entseguen plangintza egin.

Egin beharrekoak

Memoriaren azken zuzenketak egin. Artikuluaren azken bertsioa memorian txertatu.

Memoriaren azken bertsioa ADDI-ra igo. Defentsa prestatu.

Bibliografia

[1] UCAmI-2016 (IWAAL and AmIHEALTH), 2016. mami.uclm.es/ucami-2016/ [2016-ko ekainan atzitua].

[2] Reglamento de la LOPD. https://www.agpd.es/portalwebAGPD/canaldocumentacion/informes_juridicos/reglamento_lopd/index-ides-idphp.php. [2016-ko maiatzean atzitua].

[3] Etika Batzordea. http://www.ehu.eus/eu/web/ceid/home. [2016-ko maiatzean atzitua].

[4] Borja Gamecho Ibañez. A framework for abstraction and virtualization of sensors in mobile context-aware computing, 2015. https://addi.ehu.es/handle/10810/18391.

[5] Marco Besteiro y Miguel Rodríguez. Web Services., 2015. http://www.ehu.eus/mrodriguez/archivos/csharppdf/ServiciosWeb/WebServices.pdf.

[6] Orlando Fabián Brea. Manual de Web Services en PHP. http://earthcharter.org/invent/images/uploads/WEB-SERVICES-CON-PHP.pdf.

[7] Web service, 2016. https://en.wikipedia.org/wiki/Web_service.

[8] World Wide Web Consortium (W3C), 2016. http://www.w3c.es/ [2016-ko urtarrilean atzitua].

[9] Extensible Markup Language (XML), 2015. https://www.w3.org/XML/ [2016-ko otsailean atzitua].

[10] .NET - Powerful Open Source Cross Platform Development - Microsoft, 2016. https://www.microsoft.com/net [2016-ko otsailean atzitua].

[11] PHP: Hypertext Preprocessor, 2016. https://secure.php.net/ [2016-ko otsailean atzitua].

[12] Java Software | Oracle, 2015. https://www.oracle.com/java/index.html [2016-ko otsailean atzitua].

[13] Hypertext Transfer Protocol (HTTP/1.1), 2014. https://tools.ietf.org/html/rfc7230 [2016-ko otsailean atzitua].

[14] Web Services Description Language (WSDL) 1.1. https://www.w3.org/TR/wsdl.

[15] UDDI | Online community for the Universal Description, Discovery, and Integration, 2016. http://uddi.xml.org/ [2016-ko otsailean atzitua].

[16] Simple Mail Transfer Protocol, 2001. https://tools.ietf.org/html/rfc2821 [2016-ko otsailean atzitua].

[17] TCP, Transmission Control Protocol, 2012. http://www.networksorcery.com/enp/protocol/tcp.htm [2016-ko otsailean atzitua].

[18] Servicios, 2016. https://inteligencianegociositssna.wikispaces.com/Servicios [2016-ko urtarrilean atzitua].

[19] James Revelo. Servicio Web RESTful Para Android Con Php, Mysql y Json, 2015. http://www.hermosaprogramacion.com/2015/10/servicio-web-restful-android-php-mysql-json/ [2016-ko urtarrilean atzitua].

[20] Application Programming Interface, 2016. https://en.wikipedia.org/wiki/Application_programming_interface [2016-ko otsailean atzitua].

[21] JSON, 2016. http://www.json.org/ [2016-ko otsailean atzitua].

[22] Uniform Resource Locators, 2016. https://www.w3.org/Addressing/URL/url-spec.html [2016-ko otsailean atzitua].

[23] Create, read, update and delete, 2016. https://en.wikipedia.org/wiki/Create,_read,_update_and_delete [2016-ko otsailean atzitua].

[24] Burak Guzel. HTTP Headers for Dummies, 2009. http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039 [2016-ko urtarrilean atzitua].

[25] Uniform Resource Identifier (URI), 2005. https://tools.ietf.org/html/rfc3986 [2016-ko otsailean atzitua].

[26] Understanding REST Headers and Parameters, 2016. https://www.soapui.org/testing-dojo/best-practices/understanding-rest-headers-and-parameters.html [2016-ko otsailean atzitua].

[27] Media Types, 2016. http://www.iana.org/assignments/media-types/media-types.xhtml [2016-ko otsailean atzitua].

[28] Gajotres. Top 12 Best PHP RESTful Micro Frameworks (Pro/Con), 2015. http://www.gajotres.net/best-available-php-restful-micro-frameworks/ [2016-ko otsailean atzitua].

[29] Pedro Gutiérrez. Un puñado de frameworks PHP que te harán la vida más simple, 2014. http://www.genbetadev.com/frameworks/un-punado-de-frameworks-php-que-te-haran-la-vida-mas-simple [2016-ko otsailean atzitua].

[30] Web application, 2016. https://en.wikipedia.org/wiki/Web_application [2016-ko otsailean atzitua].

[31] The Apache HTTP Server Project, 2016. https://httpd.apache.org/ [2016-ko otsailean atzitua].

[32] Parham Doustdar. Re-introducing PDO – the Right Way to Access Databases in PHP, 2015. https://www.sitepoint.com/re-introducing-pdo-the-right-way-to-access-databases-in-php/ [2016-ko otsailean atzitua].

[33] Timothy Boronczyk. Migrate from the MySQL Extension to PDO, 2011. https://www.sitepoint.com/migrate-from-the-mysql-extension-to-pdo/ [2016-ko otsailean atzitua].

[34] Dejan Marjanovic. PDO vs. MySQLi: Which Should You Use?, 2012. http://code.tutsplus.com/tutorials/pdo-vs-mysqli-which-should-you-use--net-24059 [2016-ko otsailean atzitua].

[35] The HDF Group, 2016. https://www.hdfgroup.org/ [2016-ko urtarrilean atzitua].

[36] Python and HDF5 - Fast Storage for Large Data, 2012. https://youtu.be/hnhN2_TpY8g [2016-ko urtarrilean atzitua].

[37] Introduction to HDF5, 2012. http://slideplayer.com/slide/6956976/ [2016-ko urtarrilean atzitua].

[38] LAMP on Ubuntu 14.04, 2015. https://www.linode.com/docs/websites/lamp/lamp-on-ubuntu-14-04 [2016-ko urtarrilean atzitua].

[39] Slim GitHub Page, 2016. https://github.com/slimphp/Slim [2016-ko otsailean atzitua].

[40] Model-view-controller, 2016. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [2016-ko martxoan atzitua].

[41] HTML5 Tutorial, 2016. http://www.w3schools.com/html/default.asp [2016-ko martxoan atzitua].

[42] CSS Tutorial, 2016. http://www.w3schools.com/css/default.asp [2016-ko martxoan atzitua].

[43] HTML5 Website Structure, 2013. http://m5designstudio.com/wp-content/uploads/2013/01/HTML5_website_structure.jpg [2016-ko martxoan atzitua].

[44] jQuery webgune ofiziala, 2016. https://jquery.com/ [2016-ko maiatzean atzitua].

[45] PHPMailer GitHub page, 2016. https://github.com/PHPMailer/PHPMailer [2016-ko martxoan atzitua].

[46] HDF5: API Specification - Reference Manual, 2016. https://www.hdfgroup.org/HDF5/doc/RM/RM_H5Front.html [2016-ko martxoan atzitua].

[47] J. Brooke. SUS: a retrospective. Journal of usability studies, 2013. 8(2), 29-40.

[48] BITalino | DiY biosignals, 2015. http://www.bitalino.com/ [2016-ko urtarrilean atzitua].

[49] Jeff Sauro. Measuring usability with the System Usability Scale (SUS), 2011. http://www.measuringu.com/sus.php [2016ko abuztuan atzitua].

[50] International Conference on Neurorehabilitation, 2016. http://www.icnr2016.org/ [2016-ko maiatzean atzitua].