Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Fakultatea

Informatika Ingeniaritzako Gradua

▪ Gradu Amaierako Proiektua ▪

Software Ingeniaritza

Mobil aplikazio garapena web-teknologiak erabiliz

Beñat Zaldua Del Olmo

2017 - ekaina

informatika fakultatea
facultad de informática

Laburpena

Ikasleak garenean ikasketa prozesuan zerbait ez ulertzea ohikoa da. Kasu hauetan gure lagun edo klase kideei galdetzen diegu, eta azkeneko baliabide bezala, irakasleak. Hala ere, irakasleekin kontaktuan jartzea edota eraikinean aurkitzea bere tutoretza ordua ez denean zaila izan daiteke. Gainera, tutoretza orduak libre klase orduekin topo egin dezakete. Beraz, batzuetan zalantzak argitzea ez da espero bezain erreza.

Kontuan hartzen ez dugun beste jakituria iturri bat beste kurtsotako ikasleak dira. Batzuei ez zaie batere kostatzen eta beraien laguntza eskaintzeko prest daude, baina ikasle hauek nortzuk diren jakitea edota hitz egiten hastea ere arazo bat da.

Ingurune honetan garatu egin da Gradu Amaierako Lan hau. Hemen, mugikor aplikazio bat garatuko da bere helburua zalantzak dituzten ikasleak laguntzeko prest daudenekin komunikatzea dena. Aplikazio hau XXI. mendean jende gehienak eskura duen terminal batentzako garatu egin da, mugikorra hain zuzen.

Txosten honetan aplikazio hau software proiektu batekin erlazionatutako arloak landuko dira, hau da, proiektuaren planifikazioa, teknologiak eta aplikazioaren segurtasuna

Gaien Aurkibidea

Laburpena iii

Gaien aurkibidea iv

Irudi, diagramen eta taulen zerrenda ix

Kode zerrenda xiii

1. Sarrera eta motibazioa 1
2. Helburuak 3
3. Kudeaketa 5
   - 3.1. Proiektuaren irismena 5
     - 3.1.1. Burutu beharreko lanak 6
     - 3.1.2. Burutuko ez diren lanak 6
     - 3.1.3. Emangarriak 6
     - 3.1.4. Mugarriak 7
     - 3.1.5. Lanaren Deskonposaketa Egitura (LDE) 7
     - 3.1.6. Gantt diagrama 9
   - 3.2. Denboraren planifikazioa 10
   - 3.3. Arriskuak 10
   - 3.4. Komunikazio-plana 11
   - 3.5. Scrum garapen metodologia 12
   - 3.6. Kudeaketa plana 13
     - 3.6.1. Jarraipena 13
     - 3.6.2. Aldaketak 13
     - 3.6.3. Kalitatea 14
     - 3.6.4. Jarraipenaren emaitzak 14
4. Teknologiak 17
   - 4.1. Aplikazio natiboak vs. Aplikazio hibridoak 17
     - 4.1.1. Aplikazio natiboak 18
     - 4.1.2. Aplikazio hibridoak 18
     - 4.1.3. Nola aukeratu aplikazio mota 19
   - 4.2. Bezeroaren aldeko teknologiak 20
     - 4.2.1. Ionic 20
     - 4.2.2. Framework7 21

4.2.3. React Native ... 21
4.3. Zerbitzariaren aldeko teknologiak ... 21
4.3.1. CakePHP ... 22
4.3.2. Django ... 22
4.3.3. SailsJS ... 22
4.4. Aukeratuko diren teknologiak ... 22
4.5. Tresna gehigarriak ... 23
4.5.1. PhoneGap ... 23
4.5.2. Postman ... 24
4.5.3. Google Chrome ... 25
4.5.4. MySQL Workbench ... 25
4.5.5. Sublime Text 3 ... 26
5. Aplikazioaren arkitektura ... 27
5.1. Bistak ... 28
5.2. Ereduak ... 29
5.3. Kontrolagailuak ... 29
6. Diseinua ... 31
6.1. Erabilpen kasuak ... 31
6.2. Datu ereduak ... 34
6.2.1. College ... 35
6.2.2. User ... 35
6.2.3. Point ... 36
6.2.4. Petition ... 36
6.2.5. Chat ... 36
6.3. Bezeroaren pantailen diseinua ... 37
6.4. Irisgarritasuna ... 42
7. Bezero-zerbitzari arteko konexioa ... 43
7.1. Zerbitzarira eskaerak ... 43
7.2. Zerbitzariarekin komunikazioa ... 46
8. Aurkitutako zailtasunak eta erroreak ... 49
8.1. Chat zerbitzua ... 49
8.2. Zerbitzariarekin konexioa galdu ... 50
8.3. Segurtasuna aplikazioan ... 51
9. Hobekuntzak ... 53
9.1. Chat hobekuntza eta funtzionalitate berriak ... 53
9.2. Perfilak hobetu ... 55
9.3. Eskaera mota desberdinak sortu ... 55
Bibliografia ... 57
A eranskina ... 61
B eranskina ... 67
C eranskina ... 75
D eranskina ... 79
E eranskina ... 83
F eranskina ... 87

G eranskina.. 91

Irudi, Diagrama, Taula eta Kode zerrenda

Irudiak

3.1.irudia: sprint baten bizi zikloa 13

4.1.irudia: WhatsApp aplikazio natiboa 18

4.2.irudia: Tesla aplikazio hibridoa 19

4.3.irudia: PhoneGap aplikazioaren funtzionamendua 24

4.4.irudia: Postman eskaera adibidea 24

4.5.irudia: Google Chrome nabigatzailean kode arazlea mugikor pantailarekin 25

5.1.irudia: bezero-zerbitzari eredua 28

5.2.irudia: MVC egitura 28

5.3.irudia: aplikazioaren MVC egitura bezero-zerbitzari banaketarekin 30

6.3.irudia: Framework7 aplikazioaren oinarrizko egitura bat 37

6.4.irudia: Instagram, Twitter eta Hotmail aplikazioen bistak 38

6.5.irudia: Device Ready, orri bat kargatu eta orritik atzera funtzioak 39

6.6.irudia: pop-up baten definizioa 39

6.7.irudia: bistak erazagutzen 39

6.8.irudia: notifikazio adibidea 40

6.9.irudia: prompt leiho adibidea 40

7.1.irudia: eskaerak egiteko adibidea 44

7.2.irudia: JSON fitxategi bidalketa 46

7.3.irudia: eskaera bat egiteko JSON elementua 47

7.4.irudia: Ajax eskaera zerbitzarira 47

7.5.irudia: Content-Security-Policy atala 48

8.1.irudia: AES enkriptazio pausoak 51

9.1.irudia: WhatsApp eta Telegram enkriptazio mezuak 54

D.1.irudia: ebaluazio formularioaren erantzunak 81

E.1.irudia: connections.js fitxategian MySQL konfigurazioa 84

E.2.irudia: models.js fitxategiaren konfigurazioa 84

F.1.irudia: email.js fitxategiaren konfigurazioa 88

F.2.irudia: views karpetaren egitura 89

F.3.irudia: ongi etorria bidaltzeko mezuaren funtzioa User.js barruan 90

F.4.irudia: posta maketatua 90

Diagramak

3.1.diagrama: LDE diagrama 8

3.2.diagrama: Gantt diagrama 9

6.1.diagrama: erabilpen kasuak 32

6.2.diagrama: datu eredu diagrama 34

6.3.diagrama: aplikazio barruko pantailak eta nabigazioa 41

Taulak

3.1.taula: sprint-en data taula 12

3.2.taula: lan-pakete eta atazen denbora konparaketak 15

7.1.taula: eskaera helbideak funtzioaren azalpenarekin 45

B.1.taula: test erantzun taula 68

B.2.taula: test erantzun taula 69

B.3.taula: test erantzun taula 70

B.4.taula: test erantzun taula 71

B.5.taula: test erantzun taula 72

B.6.taula: test erantzun taula 73

B.7.taula: test erantzun taula 74

C.1.taula: lan pakete eta atazen denbora konparaketak 76

C.2.taula: epemugen konparaketa 76

C.3.taula: sprint-en daten konparaketa 77

Kodeak

7.1.kodea:postData JSON objektua eskaeretan 47

G.1.kodea: eskaerak automatikoki borratzeko gertaera 92

G.2.kodea: gertaeren informazioa lortzeko kodea 92

1

Sarrera eta motibazioa

Ikasleak garenean, ikasten gaudela oso ohikoa da ariketa bat nola egiten den edo teoria ondo ez ulertzea. Hau argitzekotan lagunei eta klase kideei galdetzen zaie, baina batzuetan hau ez da lagungarria edota fidagarria. Azkenean irakasleari galdetzea da irtenbide bakarra. Ala ere, batzuetan irakasleak emandako erantzuna ez da nahikoa. Ez hori bakarrik, irakaslea zentroan ordu jakin batzuetan aurkitzea ere arazo bat izan daiteke. Ondorioz, zalantza bertan gelditzen da.

Kontsideratu ez diren beste pertsona batzuk ere aurkitu daitezke ikasketa zentroan: beste kurtsoko ikasleak. Egoera berdinetik pasa diren ikasleak dira eta sortzen den zalantza beraiek ere izana posible da. Baina pertsona hauek guztiz ezezagunak izan daitezke eta laguntza eskatzea norbait ezagutu baino lehenago ez da ohikoa izaten, agian laguntzeko prest ez dagoelako.

Berarengana hurbildu beharrean, zalantza hau nonbaiten jartzea eta hau ikusten duen pertsona batek bere laguntza eskaintzea beste konponbide posible bat da. Modu honetan, ez da zalantza duena laguntza ematen duen pertsonarekin kontaktuan jartzen, alderantziz.

Arazo hau dagoeneko existitzen diren foro eta blogen bidez konpondu daiteke. Ez da zaila Internet nabigatzaile baten bidez hauek atzitzea. Zailtasuna webgunea mugikorretik kontsultak egiten direnean dator. Honen erabilera geroz eta arruntagoa da, ordenagailuarena modu ikaragarrian gaindituz. Jendeak mugikorretik nabigatzea gehiago gogoko du ordenagailu batetik egitea baino.

Gizarteak jasan duen aldaketa hau modu errazean azaldu daiteke. *Smartphone*-ak sortu zirenean, aplikazioak instalatzeko aukera sortu zen. Hauek arazo bati soluzioa ematen dieten programa txikiak dira. Nabigatzaile baten beharra alde batera gelditu egin da ondorioz. Aplikazio hauek informazioa bilatzeko, besteekin kontaktuan jartzeko edota jolasteko izan daitezke, ez dago mugarik. Ez hori bakarrik, mugikorra txikia eta leku guztietara eramateko erreza zen den terminal bat da. Dagoeneko ordenagailuak bezain azkarrak dira eta beraien artean diferentzia geroz eta txikiagoa da.

Eskaintzen duten atzipengarritasun azkartasuna eta gizartean gertatzen ari den aldaketa dela eta, aurkeztu egin den arazoari eman daitekeen soluzio bat unibertsitate mailan pertsonak komunikatzeko mugikor aplikazioa izango litzateke.

2

Helburuak

Gradu Amaierako Proiektu honek aurkeztu egin den arazoari soluzio posible bat eman nahi dion aplikazio bat garatzea izango du helburu nagusitzat. Aldi berean, honen bidez graduan zehar ikasitako kontzeptu, teknologia eta garapen metodoak nola aplikatzen diren frogatu egingo da.

Helburu pertsonal batzuk ere definitu egin dira. Tamalez, unibertsitatean sakontzen ez den eta merkatuan geroz eta gehiago eskatzen dela ikusita, mugikor teknologiak pixkanaka lantzea da hauetako bat. Honen bidez, lau urtez garatu egin diren web aplikazioek ematen duten konfort eremutik ateratzea eta mugikor teknologietan pixkanaka sartzea izango da helburuetako bat.

Bestetik, aplikazioa erabiltzaile gutxi batzuentzako soilik ez da garatu nahi, hau da, ez da pertsona jakin batzuentzako soilik sortuko. Desberdintasun hau mugikorren sistema eragilea dela eta sortzen da. Ahalik eta pertsona kantitate gehienentzako eskuragarri egotea nahi da,

modu honetan garapen bakarra eginez, plataforma anitzetarako aplikazioak nola egiten diren ikasiz.

3

Kudeaketa

Proiektua modu arrakastatsuan amaitu ahal izateko planifikazio bat garatu da proiektuaren beharrei kasu eginez. Atal honetan denborari buruzko kudeaketaz, arriskuei eta lan metodologiari buruz hitz egingo da besteak beste.

3.1 Proiektuaren irismena

Azpiatal honetan proiektuaren irismenari buruz hitz egingo da. Aurkitu daitezkeen atalen artean sortuko diren emangarriak, mugarriak eta Lanaren Deskonposaketa Egitura (LDE) aurkitzen dira.

3.1.1 Burutu beharreko lanak

Proiektuaren jarraian agertzen diren puntuak jarraituz eraman da aurrera:

- Web zerbitzari bat eta plataforma desberdinetan instalatzeko aukera emango duen aplikazioa garatuko dira.
- Aplikazioaren mantenua (sortzen diren datu hondakinak borratu) egiteko lana automatizatu egingo da. Lan hau aurrera eramateko pertsonarik ez daudenez, dena automatizatu egingo da.
- Aplikazioaren irisgarritasunaren analisi bat egingo da. Honakoa mugikor aplikazioaren inguruan eta interfazearen analisi bat egingo da.
- Garapenaren zehar sortzen diren prototipoak probatu egingo dira eta hauek dokumentatu eta aztertu egingo dira. Bukaeran, erabiltzaile talde batekin egingo da produktuaren proba bat.
- Erregistratuta dauden pertsonak eskaerak sortzeko aukera izango dute. Hauek beste erabiltzaileek ikusi ahalko dituzte eta beraien artean komunikazioa erraztuko duen mezularitza zerbitzu bat beharko du.
- Tutorial txiki bat prestatu aplikazioa nola erabiltzen den azaltzen duena.

3.1.2 Burutuko ez diren lanak

Proiektuaren irismenetik at gelditzen diren puntuak honakoa dira:

- Aplikazioak erabiltzailearen datu batzuk (datu pertsonalak eta mezuak) jaso egingo ditu eta hauek zerbitzari batean gorde egingo dira. Horrelako kasuetan LOPD (*Ley Orgánica de Protección de Datos*)[1] eta LSSI (*Ley de Servicios en la Sociedad de la Información*)[2] legedien analisi bat ekarri ohi du. Proiektu honetan honen analisia ez da jasoko.
- Mugikor aplikazioa ez da atzigarri egongo mugikorren aplikazio dendetatik. Honakoa alde batera utzi egingo da esportazioa egiteko beharrezkoa diren baliabideak eskura ez daudelako.
- Zerbitzaria ez da atzigarri egongo Internet bidez. Ez da *hosting* zerbitzu batera igoko ezta martxan jarriko. Lokalki soilik exekutatu ahalko da.
- Ez da gomendio eta funtzionamendu erroreei buruz informatzeko zerbitzurik garatuko. Horretarako korreo bat idatzi beharko da.
- Web bezeroaren garapena. Nahiz eta honetarako oinarriak garatuko diren, hau ez da garatuko.

3.1.3 Emangarriak

Proiektuak bi emangarri mota desberdin sortuko ditu:

- **Proiektuarekin zerikusia duten emangarriak:** honakoak proiektuarekin orokorrean zerikusia duten emangarriak izango dira.
    - Bilera txostenak: proiektuaren tutorearekin elkartu eta gero, hitz egin diren gaiak eta gomendazioetatik ateratzen den txostena (A eranskina).
    - Jarraipen eta kontrol txostena: iterazio bakoitza eta gero garatzen den txostena. Hemen proiektuaren jarraipena eta kontrola eramaten da (C eranskina).

- **Produktuaren barneko emangarriak:** honakoa iterazioetan sortzen diren emangarriak izango dira.
    - Test txostenak: iterazio bakoitzean garatutako funtzionalitateak probatu eta gero egindako testen erantzunak jasotzen dituen txostena (B eranskina).
    - Interfaze ebaluazio txostena: aplikazioa erabiltzaileekin probatu eta gero, egindako azterketa eta hausnarketa jasotzen duen txostena (D eranskina).
    - Kodea: garatutako produktua. Hau aparte entregatu egingo da.

- **Administrazio emangarriak:** honakoak Gradu Amaierako Proiektuarekin zerikusia duten emangarriak dira.
    - Memoria: proiektu osoa jasotzen duen dokumentua.
    - Defentsarako gardenkiak: defentsa egiten denean prestatu beharreko gardenkiak.

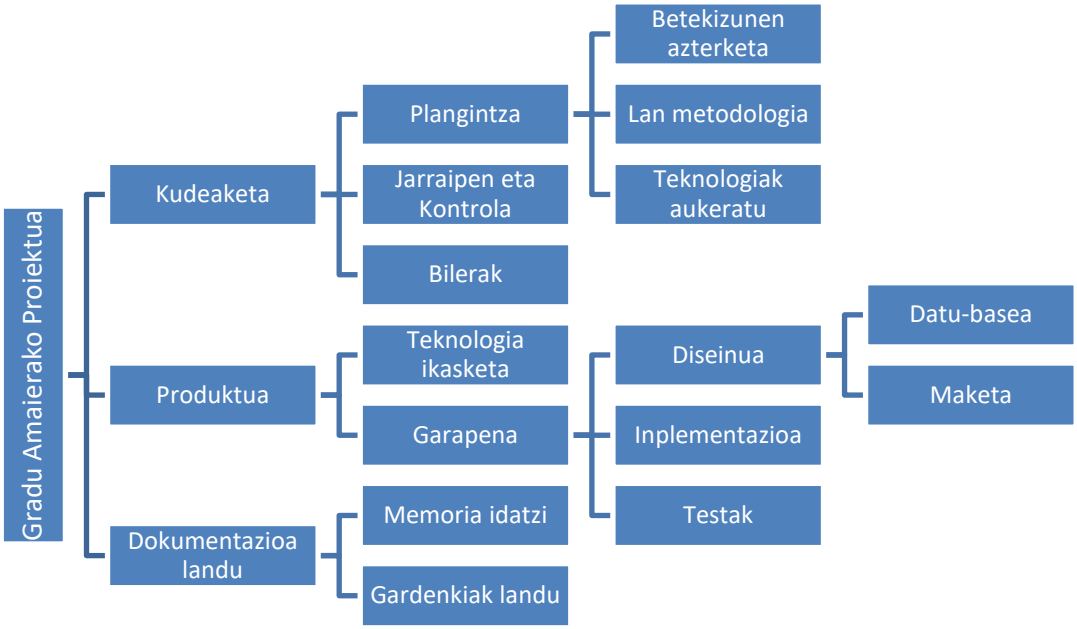
3.1.4 Mugarriak

Jarraian agertzen diren datuak proiektuan benetan garrantzitsuak diren atazak dagokien datekin izango da. Hau da, hemen ez da zehaztuko iterazio batean sortzen den dokumentuaren epemuga dokumentu hau benetan garrantzitsua ez bada.

- Proiektuaren hasiera: martxoak 21.
- Aplikazioaren lehenengo bertsioa: maiatzak 12.
- Memoriaren lehenengo bertsioa: maiatzak 12.
- Aplikazio definitiboa: maiatzak 26.
- Memoriaren azkeneko bertsioa: ekainak 9.
- Proiektuaren itxiera: ekainak 9.
- Proiektuaren defentsa: uztailak 10-13.

3.1.5 Lanaren Deskonposaketa Egitura (LDE)

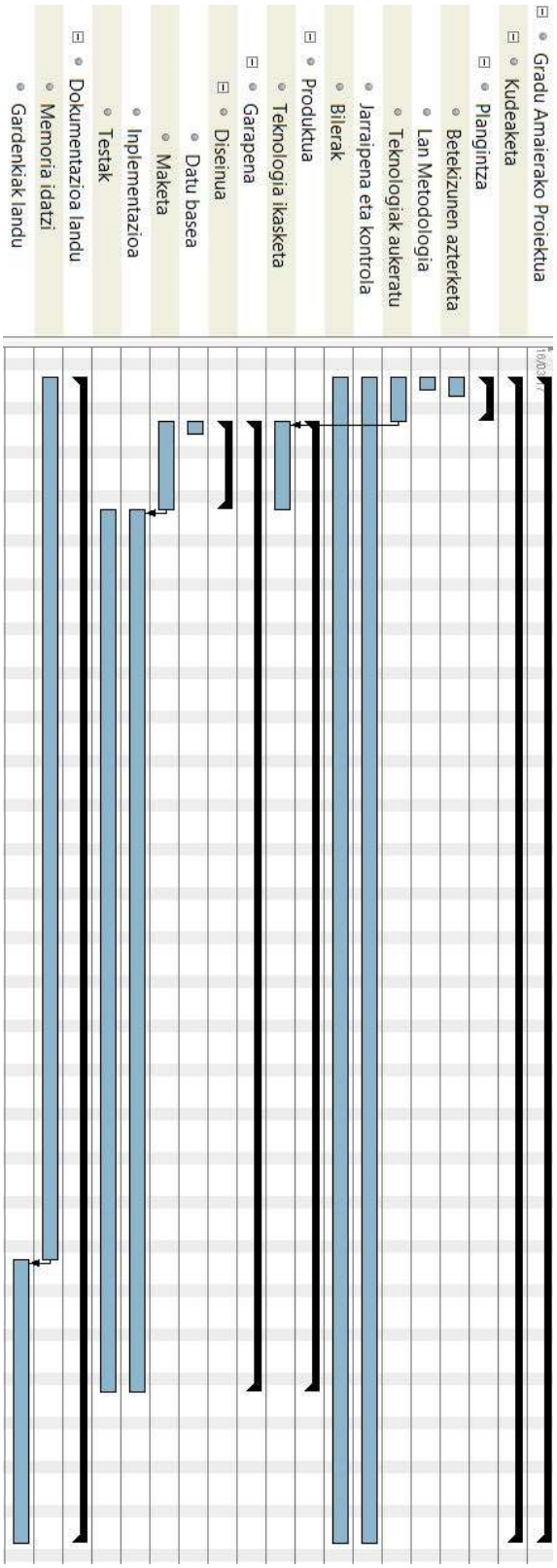
Proiektuaren kalitatezko kudeaketa bat egiteko, honen barruan dauden atazak banatu egin dira. Horretarako proiektua lan-pakete eta ataza desberdinetan banatu egin da.



3.1.diagrama: LDE diagrama

Behin lan-paketeak eta atazak definituta ditugula, hauen arteko dependentziak definitu behar dira. Proiektuen kudeaketan, 4 dependentzia mota desberdindu daitezke[3] baina kasu honetan *Finish to Start* motatako dependentziak soilik aurkitzen dira:

3.1.6 Gantt diagrama



3.2.diagrama: Gantt diagrama

3.2 Denboraren planifikazioa

Honako proiektua 300 orduko proiektu bat izango da. Jarraian, proiektua aurrera eramateko lan-pakete eta ataza bakoitzari dedikatuko zaion denboraren estimazio bat azaltzen da. Honakoa LDE diagraman ezarritako lan-pakete eta atazetatik atera dira. Proiektuaren bukaeran egindako denborak kontsultatzeko, joan C eranskinera.

- Kudeaketa: 25 or.
    - *Plangintza:* 10 or.
        - Betekizunen azterketa: 3 or.
        - Lan metodologia: 2 or.
        - Teknologiak aukeratu: 5 or.
    - *Jarraipena eta kontrola:* 10 or.
    - *Bilerak:* 5 or.
- Produktua: 175 or.
    - *Teknologia ikasketa:* 10 or.
    - *Garapena:* 165 or.
        - Diseinua: 12 or.
            - Datu basea: 2 or.
            - Maketa: 10 or.
        - Inplementazioa: 133 or.
        - Testak: 20 or.
- Dokumentazioa landu: 100 or.
    - *Memoria idatzi:* 70 or.
    - *Gardenkiak landu:* 30 or.

3.3 Arriskuak

Proiektuan hainbat arrisku izango dira kontuan izan beharrekoak:

- Gehiegizko dedikazioa: proiektu honetan denbora da baliabide garrantzitsuena. 300 ordu da muga. Garapenean desbiderapenak egotea ohikoa den gauza da, baina denboraz pasatzea izango da arriskutsua.
- Teknologiak ez funtzionatzea beraien artean: guztiz berria den ideia batekin lan egingo da, hau da, aurretik ez da aplikazio hibridorik garatu modu pertsonalean. Honakoak arrisku handia ekartzen du.
- Teknologiak eskaeretara ez moldatzea: aplikazioak izango dituen funtzionalitateak teknologiak aukeratu baino lehenago definituko dira. Ondoren, hauetara hobeto moldatzen den teknologia bat aukeratuko da baina gertatu daiteke garatu nahi den zerbait nahi den moduan garatu ezin izatea.
- Laguntza ez aurkitzea: teknologiarekin zerikusia duten zalantzak Internet bidez argitu beharko dira eta bertan aurkitzen den informazioa okerra, anbiguoa edota motz gelditu daiteke.

- **Arazo pertsonalak:** gaixotasunak, lesioak, azterketak edo pertsonarekin erlazionatutako edozein egoerak proiektuaren garapenean atzerapena ekarriko du.
- **Informazio galera:** garatzen ari den lana galtzea benetan garrantzia duen arriskuetako bat da. Ordenagailua matxuratzea, nahi gabe borratzea edota beste arrazoiren batengatik lan guztia edo partziala galdu daiteke. Honek proiektua guztiz bertan behera utz dezake.

Arrisku hauek proiektuan ahalik eta eragin txikiena izan dezaten, arriskuen-kudeaketa bat sortu da:

1. **Arazo pertsonalen kasua:** arazoren bat sortzen bada, egoera hori konpondu arte, proiektua gelditu egingo da. Honek proiektuan eragina izango du baina justifikatutako atzerapena izango da.
2. **Gehiegizko dedikazioa:** honen aurrean berriro planifikatu beharko da denbora banaketa edota funtzionalitateak ezabatu egingo dira.
3. **Teknologiak ez funtzionatzea beraien artean:** funtzionalitate baten inplementazioa bada arazoa, funtzionalitate hau alde batera utziko da eta beste bat inplementatuko da. Inplementatu nahi diren funtzionalitate gehienak arazoak emango badituzte, beste teknologia batzuk aukeratu.
4. **Laguntza ez aurkitzea:** foro, tutorialak, artikuluak irakurrita eta tutoreari galdetuta, ez bada ezer aurkitzen, fororen batean laguntza eskatu egingo da. Erantzuna jaso arte denbora bat pasa daiteke. Bitarte horretan, funtzionalitate hori etenda geldituko da eta erantzuna jasotzen denean berriz hartuko da garapena. Ez bada jasotzen, hau ezabatua izango da.
5. **Informazio galera:** hau gertatu ez dadin, kanpoko disko gogor batean eta USB memoria batean proiektuarekin zerikusia duten fitxategi guztiak gordeko dira. Makinan bertan Git erabiliko da bertsioen kontrola eramateko ere, zerbait gaizki joaten den kasuetan aurreko bertsio bat berreskuratzeko. GitHub, Bitbucket eta GitLab motatako hodei zerbitzuak alde batera uztea erabaki da, hauek erortzeko aukera dagoelako. Gainera, horrelako zerbitzuak erabiltzen direnean (nahiz eta repositorioari lizentzia eman), zerbitzu horren jabe den enpresak kode hori eskura izango du eta nolabait beraien probetxurako erabili dezakete. Hori dela eta, kopiak kanpo zerbitzuetatik at egotea erabaki da.

3.4 Komunikazio-plana

Proiektu honetan dauden interesatuak bi izango dira:

Ikaslea	Beñat Zaldua Del Olmo	bzaldua001@ikasle.ehu.es
Tutorea/Zuzendaria	Imanol Usandizaga Lombana	imanol.usandizaga@ehu.eus

Komunikazioa posta elektroniko bidez izango da. Ikasleak eskatuta, bilerak egingo dira noiz behinka, bi interesatuen artean egun bat eta ordu bat ezarriz. Bileretan egindakoa aztertu

egingo da, arazoak egon direnean aurkitutako soluzioak zein izan den (aurkitu ez bada soluzio bat bilatu) eta proiektuaren egoeraz hitz egingo da. Ikaslea arduratuko da bilera aurrera eramateaz eta hauen aktak sortzeaz, uneoro eskuragarri egongo direla proiektuan eranskin moduan (A eranskina)

3.5 Scrum garapen metodologia

Lanaren garapena aurrera eramateko, al den neurrian *Scrum* garapen metodologia[4] aplikatzea erabaki da. Honen bidez, tarteko prototipoak sortuko dira funtzionalitateak probatzeko.

Metodologiak hainbat rolen esleipena eskatzen du. Proiektuaren garapenak talde bat izatea egiten duenez, rol gehienak proiektuaren egileari esleituko zaizkio. Honakoa izango da rolen banaketa:

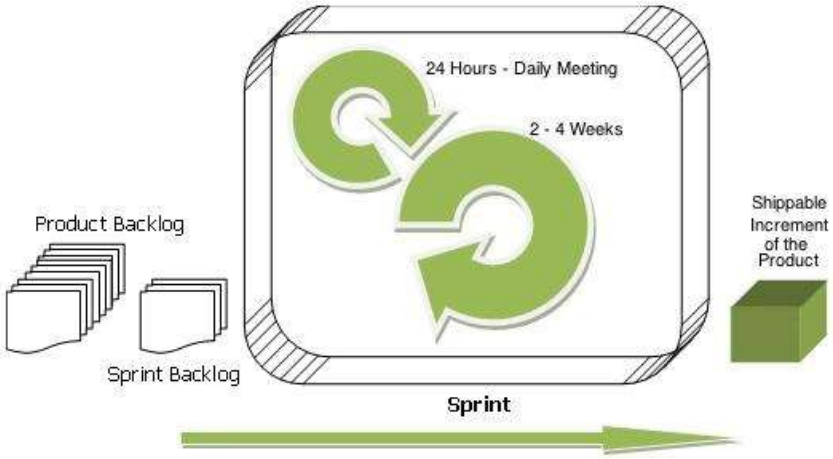
- GAP egilea: *Product Owner, Scrum Master* eta *Scrum team*.
- Tutorea: erabiltzaile lana egingo du. Nahiz eta hemendik berak onurarik ez atera, bera izango da proiektua ikusteaz, iritzia emateaz eta ideiak emateaz arduratuko den pertsona.
- Testeatzeko pertsonak: erabiltzaile lana egingo dute. Hauei aplikazio finala emango zaio probatzeko eta hauetatik hobekuntzak jasotzeko.

Tutorearekin (bezero bezala ere ezagututa) egiten diren bileretan, proiektuaren egoera zein den eta egina dagoena probatu egingo da. Bilera hauetan ematen diren iritziak eta ideiak apuntatu eta bideragarriak diren aztertuko da. Ala bada, hauek al denean inplementatu egingo dira, bestela hau ezin dela jakinaraziko zaio. Zerbait aldatzeko eskatzen bada, hurrengo *Sprint*-arekin hasi baino lehenago konpondu egingo da, bere onarpena jaso arte.

Produktuaren garapenak 8 astetan egingo da, 8 *sprint* desberdinduz. Bakoitzak aste bateko iraupena izango du. Jarraian azaltzen den taulan zehazten dira *sprint* hauentzako egin den hasiera eta bukaera data estimazioa:

Sprint izena	Hasiera data	Bukaera data
Sprint 1	03/21	03/24
Sprint 2	03/27	03/31
Sprint 3	04/03	04/07
Sprint 4	04/10	04/14
Sprint 5	04/17	04/21
Sprint 6	04/24	04/28
Sprint 7	05/01	05/05
Sprint 8	05/08	05/12

3.1.taula: sprint-en data taula



3.1.irudia: sprint baten bizi zikloa

3.6 Kudeaketa plana

Atal honetan proiektuaren jarraipena, aldaketak eta kalitatea nola kudeatuko diren azaldu egingo da.

3.6.1 Jarraipena

Proiektua garatu ahala zer eginda dagon, noiz egin den eta horretarako beharrezkoa izan den denbora jaso beharra dago. Aipatu da aurreko atalean aste batzuetan *sprint*-ak garatu egingo direla baina aste bereran garapenarekin zerikusia ez duten beste atal batzuk ere landu daitezke, adibidez dokumentazioa.

Honen kontrola eramateko, koaderno bat erabili da. Beran eguna, landu den atala, hau egiteko beharrezkoa izan den denbora eta beharrezkoa denean oharrak apuntatuko dira. Eguna bukatutzat ematen denean, guztira egindako lanorduak batu egingo dira.

Oharrak hurrengo egunean lan egiten hasi baino lehenago irakurriko diren apunteak izango dira. Zerbait erdi landuta gelditu dela edota erroreren bat aurkitu dela adierazi dezakete.

3.6.2 Aldaketak

Nahiz eta proiektu pertsonal bat izan, produktua erabiliko duten erabiltzaileen iritzia kontuan izan behar da. Kasu honetan, erabiltzaile rol hau proiektuaren tutoreak izango du. Honi egindakoa aurkeztu egingo zaio eta berak bere iritzia emateko askatasuna izango du. Proiektuan ere garatzaileak erabakita gauza berriak inplementatu daitezke.

Aldaketa hauen aurrean jarrera berdina jarraituko da. Lehenengo eskatutako edo pentsatu berri den ideia hori garatzeko bideragarritasuna aztertu beharko da. Alde batetik hori nola garatu daitekeen aztertu beharko da. Bestetik, hau aurrera eramateko beharrezkoa izango den denbora zenbatekoa izango den kalkulatu behar da.

Bideragarria den kasuetan, honakoa irismenean gehitu egingo da. Kontrako kasuetan, alde batera utziko da ideia, baina pentsatutakoa hobekuntza moduan dokumentatu egingo da.

3.6.3 Kalitatea

Softwarearen kalitatea ziurtatzeko PDCA (*Plan-Do-Check-Act*)[3] zikloa aplikatu egingo da. Honakoa garatu beharreko atal eta funtzionalitate bakoitzari aplikatuko zaio atal bakoitzean kalitate minimo bat bete dezan. Jarraian azaltzen diren puntuak kontuan izango dira:

- Funtzionalitatea modu independentean ondo funtzionatzen duen. Garatu berri dena behar bezala funtzionatzea, hau da, espero den erantzunak itzultzea kasu guztietan.
- Funtzionalitateak beraien artean ondo funtzionatzen duten. Nahiz eta modu independentean ondo funtzionatu, osotasunean ondo dabilela egiaztatu beharko da.
- Erabiltzaileentzako erabiltzeko erreza izatea, hau da, zailtasun handirik gabe eta ikasketa denbora handia eskatu gabe, erabiltzeko erraztasuna izatea.
- Mantenua erreza izatea. Aplikazioan zerbait aldatu nahi bada, honi ez eragitea edo ahalik eta eragin ahulena izatea.
- Egokitzeko gaitasuna. Kontuan izan behar da non erabiliko den, kasu honetan mugikorretan. Sistema eragile eta pantaila tamaina desberdinak egongo dira, beraz guztietan modu egokian ikusten dela ziurtatu beharko da.

Aldiz, beste atal batzuk ez dira hain kontuan izango:

- Aplikazioa segurua den. Nahiz eta datu multzo batzuekin lan egingo den, hau ez da hain kontuan izango.
- Fidagarritasuna eta eraginkortasuna.

3.6.4 Jarraipenaren emaitzak

Aipatu den bezala, proiektua 300 ordu bete beharko ditu. Ataza eta lan-paketeen banaketa bat egin da eta bakoitzari denbora hori betetzeko denbora estimazio bat esleitu zaio. Jarraian, proiektuaren atal nagusietan egindako denbora erreala azaltzen da. Informazio gehiago C eranskinean.

LAN PAKETEA/ATAZA	ESTIMATUTAKO ORDUAK	ERABILITAKO ORDUAK	DESBIDERAKETA
Kudeaketa	**25 h**	**21h 30'**	+3h 30'
Plangintza	10 h	8h 30'	+1h 30'
Betekizunen azterketa	3 h	3h	-
Lan metodologia	2 h	1h 30'	+30'
Teknologiak aukeratu	5 h	4h	+1h
Jarraipena eta kontrola	10 h	10h 30'	-30'
Bilerak	5 h	2h 30'	+2h 30'
Produktua	**175 h**	**176h 55'**	-1h 55'
Teknologia ikasketa	10 h	15 h	-5h
Garapena	165 h	161h 55'	+3h 5'
Diseinua	12 h	13h	-1h
Datu basea	2 h	1 h	+1h
Maketa	10 h	12 h	-2h
Inplementazioa	133 h	131 h 55'	+1h 5'
Testak	20 h	17 h	+3h
Dokumentazioa landu	**100 h**	**94 h 40'**	+5h 20'
Memoria idatzi	70 h	64 h 40'	+5h 20'
Gardenkiak landu	30 h	30 h	-
Guztira:	300 h	293h 5'	+6h 55'

3.2.taula: lan-pakete eta atazen denbora konparaketak

Proiektuaren denbora totala ikusita, desbideraketa oso txikia izan dela ikusi daiteke, beraz denboren estimazioa nahiko zehatzak izan direla esan daiteke. Ala ere, desbiderapen batzuk aurkitu daitezke, adibidez produktuan gertatu direnak. Honakoa teknologien ikasketa dela eta izan da. Ezagutzen ez ziren teknologiak aukeratu direnez, hauek nola funtzionatzen duten, egitura zein den eta dokumentazioa irakurri behar da, hortik denboraren desbideraketa. Bestetik, maketa bat garatzea erabaki zen. Maketaren diseinuan ere aurkitu daiteke desbideraketa garrantzitsu bat. Honakoaren bidez bukaerako produktuak izango zuen itxura nolakoa zen lortu nahi izan da.

Teknologiaren ikasketaren desbiderapenari dagokionez, proiektuaren gain ez du eragin handirik izan. Honakoa produktua garatzen baino lehenago egin beharreko pausoa zenez, garapenerako estimatutako orduak ondoren soberan izatea espero zen (ala izan da). Maketarekin berdina gertatu da. Garapen prozesuan maketa prest izateak ordu batzuk kentzen dizkio honi.

Ondorioz, gertatutako desbideratze hauen aurrean ez da konponbide jakin bat hartu jasangarriak izan diren desbideraketak baitziren.

4

Teknologiak

Atal honetan bezeroaren eta zerbitzariaren atalak garatzeko dauden aukeren analisi bat egingo da eta beharretara hobeto egokitzen dena aukeratu egingo da.

Bezeroaren aldea garatzeko dauden aukeren analisi sakon bat ere egingo da. Zehatzagoak izanda, modu natiboan eta hibridoaren arteko diferentzia zein den eta erabakia ere agertu egingo da.

Behin analisia eginda, garapenean erabiliko diren tresnak aipatu egingo dira eta kasu bakoitzean zertarako erabiliko diren.

4.1 Aplikazio natiboak vs. Aplikazio hibridoak

Mugikor aplikazio bati buruz hitz egiten denean bi motatako aplikazioak garatu daitezkeela azaldu beharra dago[5]. Aplikazio hauek natiboak edo hibridoak izan daitezke eta beraien

arteko garapen prozesua guztiz desberdina da, horregatik garatzen hasi baino lehenago proiektura hobeto egokitzen den mota aukeratu beharko da.

4.1.1 Aplikazio natiboak

Aplikazio natibo deritzen aplikazioak mugikorrek duten sistema eragilearentzako bereziki garatuta eta optimizatuta daude. Mota honetako aplikazioak mugikorren dituzten osagaiak (kamera, GPS, giroskopioa, …) modu errazean erabiltzea ahalbidetzen dute batik bat. Hauen garapena egin ahal izateko programazio lengoaia jakin batzuk erabili behar dira, adibidez Android sistema eragilea erabiltzen duen mugikor batentzako Java ikasi behar da (bezero zatia garatzeko gutxienez).

Garapenari dagokionez, denbora eta diru kostua handia ekarri ohi du. Ala ere, hemendik lortzen diren produktuak errendimendu ona izan ohi dute. Erabiltzaileak lortzen duen funtzionamendu esperientzia ere oso ona dela aipatu behar da.

4.1.irudia: WhatsApp aplikazio natiboa

4.1.2 Aplikazio hibridoak

Aplikazio hibridoak web aplikazio eta aplikazio natibo baten arteko nahasketa bat da eta aplikazio natiboekin bezala, instalazioa dendatik egiten da. Mota honetako aplikazioak jaurtitzen direnean, aplikazio bat jaurtitzen denean ikusten dugun efektu berdina ikusiko da baina atzetik, nabigatzaile bat irekitzen da eta bertan exekutatuko da aplikazioa. Beraz, nahiz eta mugikor aplikazio moduan instalatu izana, web aplikazio bat exekutatzen dago. Mota honetako garapenak ere mugikorrak dituen hainbat osagai erabiltzeko aukera ematen du baina murriztapenekin aurkitu gaitezke. Askotan hau egin al izateko liburutegi eta osagai gehigarriak gehitu behar zaizkio kodeari.

Natiboetan ez bezala, hauen garapena azkarra, merkea eta ez da sistema eragilearen menpekoa. Sistema eragilearen menpeko ez izateak garatzaileentzako abantaila bat ekartzen

du, kodea behin garatuta, merkatuan dauden mugikor sistema eragile guztietan exekutatu ahalko baita.

Programazio lengoaiei dagokienez, HTML (*HyperText Markup Language*), CSS (*Cascading Styleshit)* eta JavaScript lengoaietan oinarritzen dira gehienak (edo hauen aldaera batean). Java, C eta antzeko programazio lengoaietan ez bezala, hauek oso azkar ikasten diren lengoaiak dira. Web teknologiekin garatutako proiektuak gainera ugariak dira, webgune guztiek erabiltzen baitituzte, beraz oso erreza izaten da funtzionalitate bat dagoeneko Interneten eginda aurkitzea.

4.2.irudia: Tesla aplikazio hibridoa

4.1.3 Nola aukeratu aplikazio mota

Aplikazio natiboa edo hibridoa aukeratzeko aldez aurretik jakin beharreko informazioaren artean honakoa aurkitu behar da:

- Garatzeko dagoen denbora.
- Proiektuarentzako dagoen aurrekontua.
- Programazio lengoaiak ezagunak diren.
- Plataforma anitzetarako izango den.

Proiektu honi dagokionez, ez da inongo aurrekonturik egongo eta denbora aldetik, ez da asko izango (1-3 hilabeteko tartea egon ohi da) eta zenbat eta pertsona gehiagorentzako

atzigarri egon, orduan eta hobeto. Programazio lengoaiei dagokionez, ez da arazorik sortzen kasu honetan.

Eman dezagun aplikazio modu natiboan garatu egingo dela eta bai Android eta iOS sistemetarako eskuragarri egongo dela. Aplikazioak garatzen direnean erabiliko diren lengoaiak guztiz desberdinak izango dira, baina erabiltzaileek aplikazio berdina behar dutela ikusi beharko dute, beraz honen gainetik diseinu lan oso garrantzitsua dago. Honekin esan nahi dena Android aplikazioa irekitzen denean eta iOS aplikazioa irekitzean ez direla desberdintasunik egon behar erabiltzailearentzako da. Gainera, aplikazioak bezeroen arteko komunikazioa ahalbidetzen badu, zerbitzari bereziak behar dira hau egiten ahalbidetzen dutenak.

Aipatu egin diren arrazoiengatik, proiektura hobeto moldatzen den aplikazio mota aplikazio hibridoa izango da.

4.2 Bezeroaren aldeko teknologiak

Garatuko den aplikazioa hibridoa izango dela aipatu da aurreko puntuan. Horrelako aplikazioak garatzeko *framework* deritzen lan inguruneak erabiltzen dira, beharrezkoak diren liburutegi eta fitxategiak dagoeneko bertan aurkitzen baitira.

Horrelako aplikazioak garatzen ahalbidetzen dituzten lan inguruneak geroz eta gehiago dira baina Interneten gehien aipatzen direnak aztertuko dira hemen: **Ionic**[6], **Framework7**[7] eta **React Native**[8].

4.2.1 Ionic

Google enpresak sortutako AngularJS programazio lengoaian oinarritzen den lan ingurunea da. MVC egitura jarraitzen duten aplikazioak garatzen ahalbidetzen du eta eskaintzen dituen diseinu baliabideei esker, lortzen diren aplikazioak natiboen itxuratik oso gertu gelditzen dira.

Lan ingurune honek eskaintzen digun dokumentazioa ugaria da eta honekin garatutako proiektuak aurkitzea erreza da. Komunitateari dagokionez, jende dezentek sortzen du, beraz arazoren batekin aurkituz gero, Interneten dauden foroetan ez da zaila erantzun bat aurkitzea.

Zerbitzu gehigarri batzuk ere ematen ditu lan ingurune honek. Hauen artean aplikazioak modu azkarrean garatzeko ingurunea, aplikazioa ikuskatzeko aplikazioa, etab. Zerbitzu gehigarri hauek lagungarriak izan daitezke garapenean, baina hauek erabiltzeko dependentzia ere sortu daiteke.

4.2.2 Framework7

HTML, CSS eta DOM7 (JavaScript aldaera bat, jQuery-ren antz handia duena) teknologietan oinarritzen den lan ingurunea da. Eskaintzen duen diseinu baliabideei esker, aplikazioa iOS (iPhone sistema eragilea) edo Material Design (Android sistema eragilea) bezala diseinatu daiteke aplikazioa. Honi esker, lortzen diren aplikazioak natiboetan lortzen den itxuratik oso gertu gelditzen da.

Garapenerako eskaintzen duten dokumentazioa ugaria da. Ez hori bakarrik, honekin garatu daitezkeen aplikazioen adibideak eskaintzen dira ere, hauen kodea kontsultatzeko aukerarekin. Komunitateari dagokionez, oso handia da ere. Komunitate honek *plugin*-ak sortzen ditu lan ingurune honetarako eta hauek proiektuan instalatzea lan erreza da.

Aplikazioa garatzeko oinarrizko egitura ematen du soilik, beraz aplikazio gehigarriak behar dira honen esportazioa eta simulazioak egiteko. Gomendatzen den aplikazioa Adobe-ren *PhoneGap* aplikazioa da.

4.2.3 React Native

Facebook enpresak garatutako lan ingurunea. Honakoa, beraiek sortutako React eta JavaScript lengoaietan oinarritzen da. React erabiltzaile interfazeak sortzeko balio du eta JavaScript lengoaiaren aldaera bat da.

Nahiz eta nahiko berria izan teknologia hau, komunitate oso handia du. Aurkitu daitezkeen proiektuak anitzak dira baina dokumentazioa ez da hain ugaria. Ala ere, egunetik egunera hau handitzen doa.

4.3 Zerbitzariaren aldeko teknologiak

Zerbitzari baten garapenerako dauden aukerak aztertu baino lehenago, aipatu behar da aldez aurretik web zerbitzari bat baliatuz garatuko dela zerbitzaria. Horrela, aurrerago lan honen garapenarekin jarraitu nahi bada web aplikazioa gehituz, zerbitzaria dagoeneko garatuta egongo da eta berrerabiltzeko aukera bertan dago.

Zerbitzari bat garatzeko teknologiak asko dira. Proiektu honetan interesatzen zaigun zerbitzariak MVC egitura bat jarraitu behar du, beraz egitura hau garatzen onartzen dituen lan inguruneak bilatu egin dira. Beraien artean guztiz desberdinak diren programazio lengoaietan oinarritzen diren hiru ingurune aukeratu dira: **CakePHP**[9], **Django**[10] eta **SailsJS**[11].

4.3.1 CakePHP

PHP programazio lengoaian oinarritzen den lan ingurunea. MVC egitura jarraitzen du, objektuei orientatuta dago eta bere kontsolaren laguntzarekin aplikazioak oso azkar inplementatu daitezke.

Garatu beharrekoa eskuz edo Interneten aurkitzen diren *plugin*-en bidez egin daiteke. *Plugin* hauen artean fitxategiak igotzeko *plugin*-a edo erabiltzaile baten autentifikazio (sesio hastea, ixtea, etab.) aurkitzen dira.

Dokumentazio eta komunitate handia duen ingurunea da. Komunitatea handiari esker, aplikazioarentzako *plugin*-ak aurkitzea ere lan erreza da, gehienak GitHub-en aurkitzen dira. Beste batzuk dokumentazioan bertan gomendatzen dira deskargatzeko iturriarekin.

4.3.2 Django

Python programazio lengoaian oinarritzen den ingurunea. Honakoak MVC egitura bat jarraitzen du (MTV, *Model-Template-View* bezala ezaguna) eta objektuei orientatuta dago ere. Python programazio lengoaian oinarritzen denez, gutxi kodetzearekin, funtzionalitate oso eraginkorrak lortzen dira, garapena oso azkarra eginez.

Dokumentazio eta komunitate handiko ingurunea da eta honakoa erabilita aurkitzen diren proiektuak anitzak dira.

4.3.3 SailsJS

JavaScript programazio lengoaian oinarritzen da eta NodeJS ingurunearen aldaera bat da. Honakoak MVC egitura jarraitzen du, NodeJS ez bezala.

NodeJS-k eskaintzen dizkigun moduluak ere instalatu daitezke, beraz zerbitzu asko dagoeneko garatuta daude komunitate handiari esker. Lan ingurune honetan zerbait nola garatzen den aurkitzen ez bada, beti NodeJS-ren dokumentazioa begiratu daiteke, baina erreferentzia bezala soilik.

Eskaintzen dituen *blueprint*-ei esker, kodearen egitura modu automatikoan garatu daiteke (kontrolagailu eta ereduen atalak).

4.4 Aukeratuko diren teknologiak

Garatu egingo den aplikazioak datu base baten beharra izango du datuak gordetzeko. Bi datu base mota garatu daitezke: erlazionala edo ez erlazionala. Kasu honetan, ez erlazionala

den datu base bat garatzeak ez du inongo zentzurik, datuen arteko erlazio bat egongo delako. Datu base erlazionalen artean MySQL da beste proiektuetan gehien erabili dena, hori dela eta hau erabiltzea erabaki da.

Bezeroaren zatia garatzeko aurkeztu egin diren inguruneetatik. React Native ezagutzen ez den React erabiltzen du eta hau ikasteak denbora asko eramango luke. Ionic-ek ere arazoak ekar ditzake, beraien zerbitzu gehigarri horiei garrantzia ematen hasi baitira eta hauen erabilera behartzeak kezka sortzen du.

Hau dela eta, bezeroaren zatia garatzeko Framework7 erabiltzea erabaki da. Guztiz doakoa den ingurunea da eta PhoneGap aplikazioaz aparte ez du zerbitu gehigarririk eskatzen. Gainera dagoeneko ezagutzen diren teknologiak erabiltzen ditu garapena egiteko. Honi esker, plataforma anitzetarako diren aplikazioak garatu daitezke zailtasun handirik gabe. Ez hori bakarrik, nahiz eta HTML, CSS eta JavaScript lengoaietan oinarrituta egon, mugikorretara bideratuta dago lan ingurune hau. Beraz, nahiz eta lengoaiak ezagutu, egoera berri batean nola erabili ikasi beharko da.

Zerbitzariaren zatia garatzeko SailsJS erabiltzea erabaki da. Honek MySQL motako datu basearekin ondo egiten du lan eta aukeratutako bezero lan inguruneak JavaScript erabiltzen duenez, dena ondo lotu daiteke inongo arazorik gabe. CakePHP eta Django inguruneek konfigurazio luzea ekartzen du instalatu beharreko pakete eta programengatik, SailsJS-rekin gertatzen ez dena.

Bezeroarentzako eta zerbitzariarentzako aukeratutako teknologiak JavaScript lengoaia dute komuna. Honi esker, badakigu bi aldeen arteko komunikazioa eta funtzionamendua posible dela.

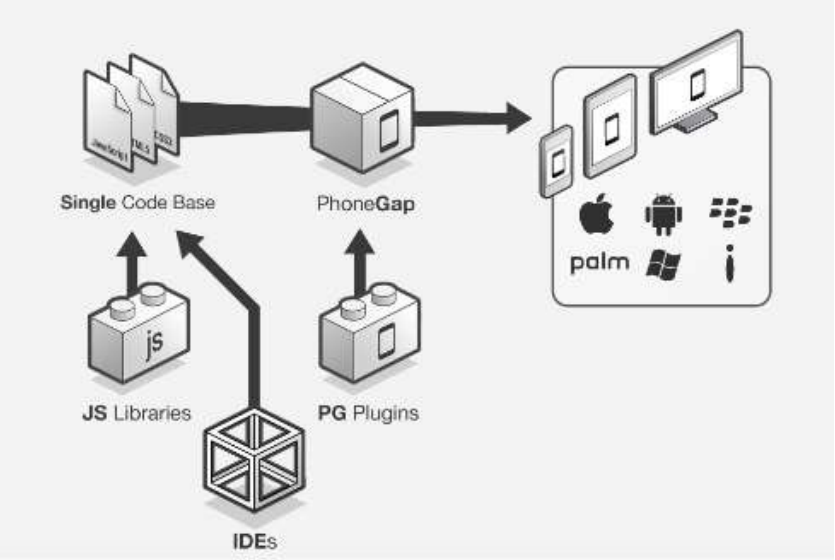
4.5 Tresna gehigarriak

Aukeratutako teknologiaz gain, beste erremienta batzuk erabiliko dira garapen prozesurako eta garatutako funtzionalitateak probatzeko.

4.5.1 PhoneGap

Programa hau[12] aplikazio hibridoak ordenagailuan martxan jarri daitezke, bere itxura nola gelditu den ikusteko eta funtzionalitateak probatzeko. Lan ingurune bat ere bada, beraz bertan ere aplikazio hibridoak garatu daitezke.

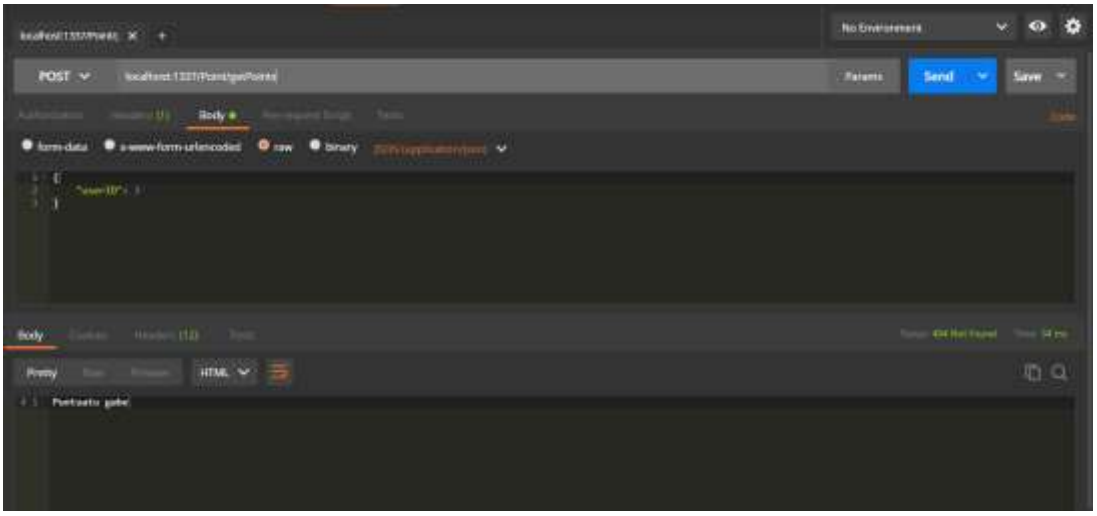
Aplikazio honek garatutako kodea mugikor aplikazio bihurtuko du. Horretarako sortutako kodea eman beharko zaio. Gainera, PhoneGap berak *plugin* edo osagaiak gehitzeko aukera ere ematen du. Honen bidez, gure aplikazioa zabaldu egingo da eta aukeratutako lan inguruneak eskaintzen ez dituen funtzionalitateak gehitu daitezke.



4.3.irudia: PhoneGap aplikazioaren funtzionamendua

4.5.2 Postman

Web zerbitzuak probatzen ahalbidetzen digun HTTP (*HyperText Transfer* Protocol) bezero aplikazioa[13] da. Honi zein web zerbitzuaren URL-a (*Uniform Resource Locator*) zehaztu, zein HTTP metodo (POST, GET, …) erabili nahi den eta beharrezkoa denean, datuak adierazi behar zaizkio. Ondoren dagokion erantzuna pantailan erakutsiko du.

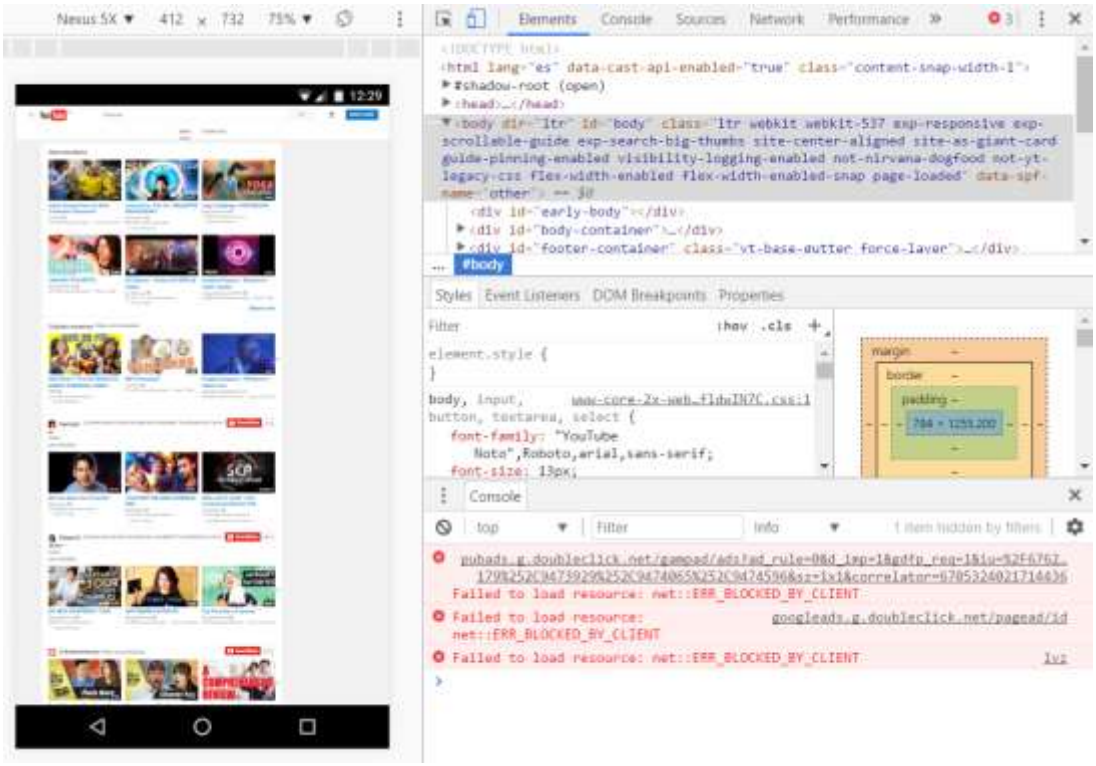


4.4.irudia: Postman eskaera adibidea

4.5.3 Google Chrome

Nabigatzaile bat izateaz gain, erremienta oso boteretsua da. Aplikazioaren kodea araztu daiteke eta pantaila desberdinetan nola ikusiko den simulatu daiteke (mugikor pantailak eskura daude).

Ez hori bakarrik, mugikor bat konektatzen bada ordenagailura eta bertan aplikazio hibrido bat jaurtitzen bada, bertan gertatzen dena ere arazteko aukera ematen digu nabigatzaileak. Bere barnean aplikazioak instalatu daitezke, bertan egin daitezkeen gauzak zabalduz. Aplikazio hauek garatzaileentzako (Postman, kodea arazteko aplikazioak, …) edota erabiltzaileentzako (AdBlocker, Angry Birds jokoa, …) aplikazioak izan daitezke.



4.5.irudia: Google Chrome nabigatzailean kode arazlea mugikor pantailarekin

4.5.4 MySQL Workbench

Datu base bat sortzeko eta maneiatzeko erabiltzen den aplikazioa. Aplikazioak datu basean gordetzen dituen datuak aztertzeko erabiliko da. Sistema eragile desberdinetarako eskuragarri dago eta guztiz doakoa da.

Funtzionalitate oso interesgarriak eskaintzen ditu, gertaerak sortzeko, datu baseak migratzeko eta datu basea diseinu bisual batean ikusteko aukerak batik bat.

4.5.5 Sublime Text 3

Testu editore oso boteretsua. Kodea editatzeko oso egokia den erremienta, programazio lengoaia asko baitaude eskura hemen. Testua kolore desberdinez nabarmentzen du edizioa errazagoa egiteko eta, programazio lengoaiaren arabera, kodea modu automatikoan idatzi dezake ere.

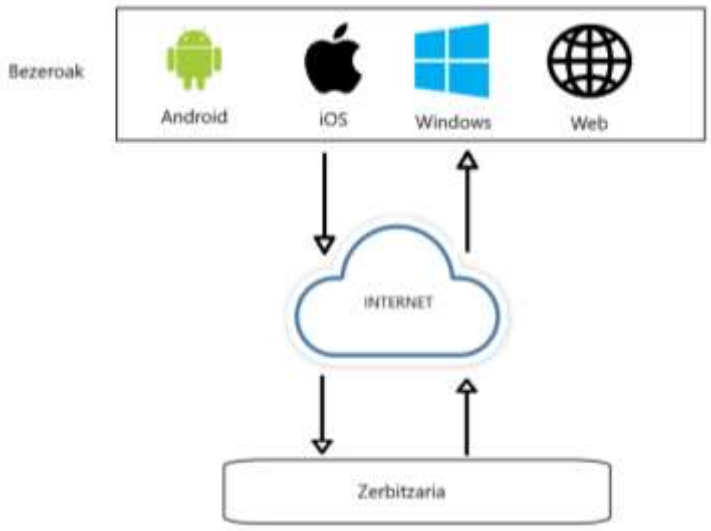
5

Aplikazioaren arkitektura

Lan honetatik mugikor aplikazio bat aterako dela aipatu egin da eta *thin client*[14] moduan garatu nahi da bezeroaren aldea. Hau bezeroaren aldean logika ahalik eta gehien gutxitzea ekartzen du. Ondorioz zerbitzari baten dependentzia sortuko da. Beraz, bi zati desberdin garatu beharko dira, bezeroa eta zerbitzaria.

Bezeroa erabiltzaileek beraien mugikorretan instalatu beharko duten aplikazioa izango da, *front end*-a izango dena. Horrelako garapena ahalbidetzen duten teknologiak ugariak dira gaur egun baina bat aukeratu baino lehenago hainbat atal aztertu behar dira, aplikazioa natiboa edo hibridoa izango den adibidez.

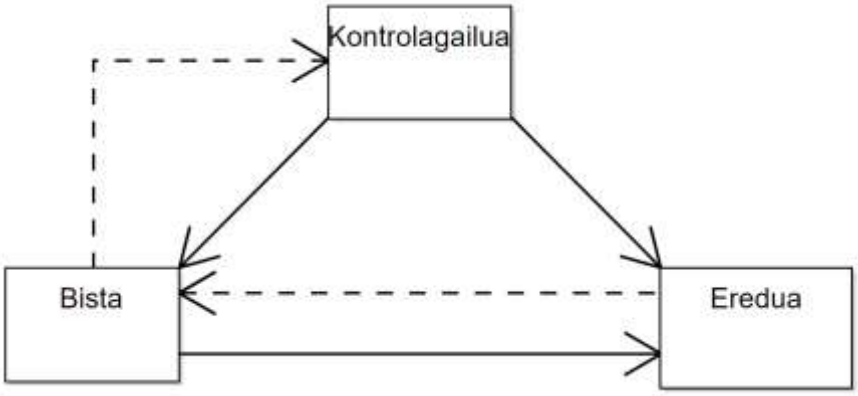
Zerbitzariarekin, *back end*-a, berdina gertatzen da. Zerbitzari bat modu askotan garatu daiteke eta egoerarako bakoitzerako egokiena zein den aztertu beharko da lehenengo. Kasu honetan, aldez aurretik erabaki da web zerbitzari bat garatzea, horrela proiektuarekin jarraitu nahi bada eta webgune bat sortu nahi bada, oinarria garatuta egongo da.



5.1.irudia: bezero-zerbitzari eredua

Bi atalak konektatzeko Internet erabili beharko da. Bezeroek zerbitzariari eskaerak egingo dizkio eta zerbitzariak dagokion erantzuna itzuli egingo du kasu bakoitzean.

Bezero-zerbitzari arkitekturarekin batera, MVC (*Model-View-Controller*)[15] arkitektura ere erabiliko da. Egitura honek datuen eredua erabiltzaile interfazetik eta logikatik banatzea proposatzen du, mantenua eta kode berrerabilpena erraztuz, kodea 3 geruzatan banatzen baitu.



5.2.irudia: MVC egitura

Bi arkitekturak konbinatzeko, bezero-zerbitzariko atal bakoitza MVC arkitekturan dagoen geruza batekin parekatu beharko da.

5.1 Bistak

Bista (*view*) interfaze grafikoetaz eta kontrolagailuak eta ereduak emandako datuak modu egokian adierazteaz arduratzen eta erabiltzaileekin elkarreragingo duen geruza izango da.

Bezero-zerbitzari egiturarekin erlazionatu behar badugu, honi dagokiona bezero atala izango da. Erabiltzaileek beraien mugikorretan aplikazioa instalatu egingo dute, beraz ez da logikarik beharko.

Honakoa garatzeko bezeroaren aldea garatzeko erabakitako teknologia erabiliko da, Framewok7 zehazki. Beraz, atal hau HTML, CSS eta JavaScript lengoaien bidez garatu egingo da.

5.2 Ereduak

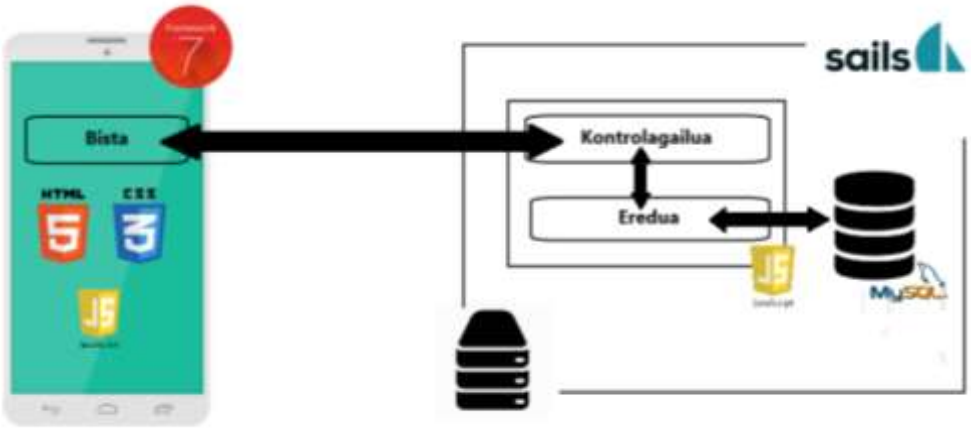
Eredua (*Model*) datu basearen adierazpen bat izango da. Hemen datu basean gordeko diren datuak zein datu motakoak izango diren, izango duten luzera, taulen arteko erlazioak, etab. definitu beharko da. Gorde edo eguneratu nahi diren datuak dagokien eredua jarraitzen dela atal honetan aztertzen da. Muinean, honakoa datu baseko taula baten diseinua izango da eta bere lana datu basearekin elkar eragitea izango da.

Bezero-zerbitzari egituran, honakoa zerbitzarian aurkituko dugun geruza izango da, bertan aurkitzen baita datu basea. Honakoa garatzeko JavaScript lengoaia erabiliko da eta MySQL datu basearekin elkarreragingo du.

5.3 Kontrolagailuak

Kontrolagailua (*Controller*) aplikazio baten logika guztiaz arduratzen den geruza da. Hemen erailtzaileengandik eskaerak edo ereduetatik datuak jasoko dira, hauetan funtzio batzuk aplikatuko dira erantzun egokia prestatuz, eta ondoren erantzun hau dagokion lekura bidaliko du.

Berriro ere, bezero-zerbitzari arkitekturan, honakoa zerbitzariaren aldean garatu beharreko atal bat izango da. Hau ere bezeroan garatzea dago, baina *thin client* bat garatzea erabaki denez, zerbitzariaren aldean jarriko da.



5.3.irudia: aplikazioaren MVC egitura bezero-zerbitzari banaketarekin

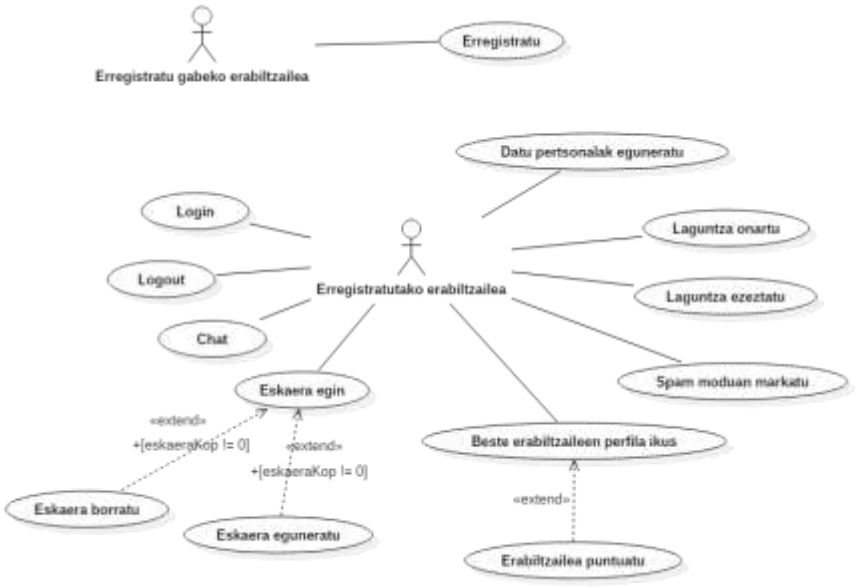
6

Diseinua

Atal honetan aplikazioaren diseinuaz hitz egingo da. Aurkitu daitezkeen atalen artean, aplikazioak izango dituen erabilpen kasuak, datu ereduak eta bezeroaren aldeko pantailen diseinua eta honen irisgarritasunaren azterketa bat egongo dira.

6.1. Erabilpen kasuak

Erabiltzaile batek prozesu bat aurrera eramateko jarraitu beharreko pausoak azaltzen dira atal honetan. Aplikazio honek bi aktore mota izango ditu: erregistratu gabe dagoen erabiltzailea eta erregistratutako erabiltzailea. Aktore bakoitzak erabilpen kasu jakin batzuk ahal ditu aurrera eraman.



6.1.diagrama: erabilpen kasuak

ERABILPEN KASUEN DESKRIBAPEN LABURRA:

ERREGISTRATU GABEKO ERABILTZAILEA

Erregistratu

- Laburpena: erabiltzaileak bere datuak emango ditu, sisteman gordeko direnak, eta sesioa hasteko baimena izango du.
- Aurrebaldintza: sartutako email-a errepikatuta ez egotea.
- Postbaldintza: erabiltzaile erregistratu bihurtzen da erabiltzailea.

ERREGISTRATUTAKO ERABILTZAILEA

Login

- Laburpena: aplikazioan sartzeko eta sesioa hasieratzeko.
- Aurrebaldintza: sesioa hasita ez egotea.
- Postbaldintza: sesioa hasten da.

Logout

- Laburpena: sesioa itxi.
- Aurrebaldintza: sesioa hasita izan.
- Postbaldintza: sesioa itxi egingo da.

Eskaera egin

- Laburpena: eskaera berri bat egingo da eta sisteman gordeko da.

- Aurrebaldintza: saioa hasita izan.
- Postbaldintza: eskaera berri bat sortuko da.

Eskaera eguneratu

- Laburpena: egindako eskaera baten datuak aldatu edo eguneratu.
- Aurrebaldintza: sesioa hasita izan eta eguneratu nahi den eskaera existitzea.
- Postbaldintza: eskaeraren datuak eguneratu egiten dira.

Eskaera borratu

- Laburpena: eskaera pertsonal bat borratu.
- Aurrebaldintza: sesioa hasita izan eta borratu nahi den eskaera aurretik existitzea.
- Postbaldintza: berarekin erlazionatutako guztia ezabatu egingo da.

Spam moduan markatu

- Laburpena: beste pertsona batek egindako eskaera spam edo ezegoki bezala markatu.
- Aurrebaldintza: sesioa hasita izan.
- Postbaldintza: aplikazioan berriz sartzean eskaera hori ez da ikusiko.

Datu pertsonalak eguneratu

- Laburpena: erabiltzaileak erregistro orduan sartutako aldatzeko edo eguneratzeko.
- Aurrebaldintza: sesioa hasita izan.
- Postbaldintza: datuak eguneratu egingo dira.

Chat

- Laburpena: beste ikasle batekin kontaktuan jartzeko. Honen bidez beste erabiltzaile batekin mezuak bidaltzeko aukera dago.
- Aurrebaldintza: sesioa hasita izan.
- Postbaldintza: chat berri bat sortuko da bien artean.

Laguntza onartu

- Laburpena: beste erabiltzaile batekin hitz egin ondoren laguntza onartu egingo da. Berarekin hitz egingo da laguntza hori nola eraman aurrera.
- Aurrebaldintza: sesioa hasita izan.
- Postbaldintza: onartutako eskaera ez da ikusgai egongo beste erabiltzaileentzako baina chat-a mantenduko da.

Laguntza ezeztatu

- Laburpena: beste erabiltzaile batekin hitz egin ondoren laguntza ezeztatzea erabaki da akordiorik egon ez delako.

- Aurrebaldintza: sesioa hasita izan.
- Postbaldintza: chat-a ezabatu egingo da.

Beste erabiltzaileen perfila ikusi

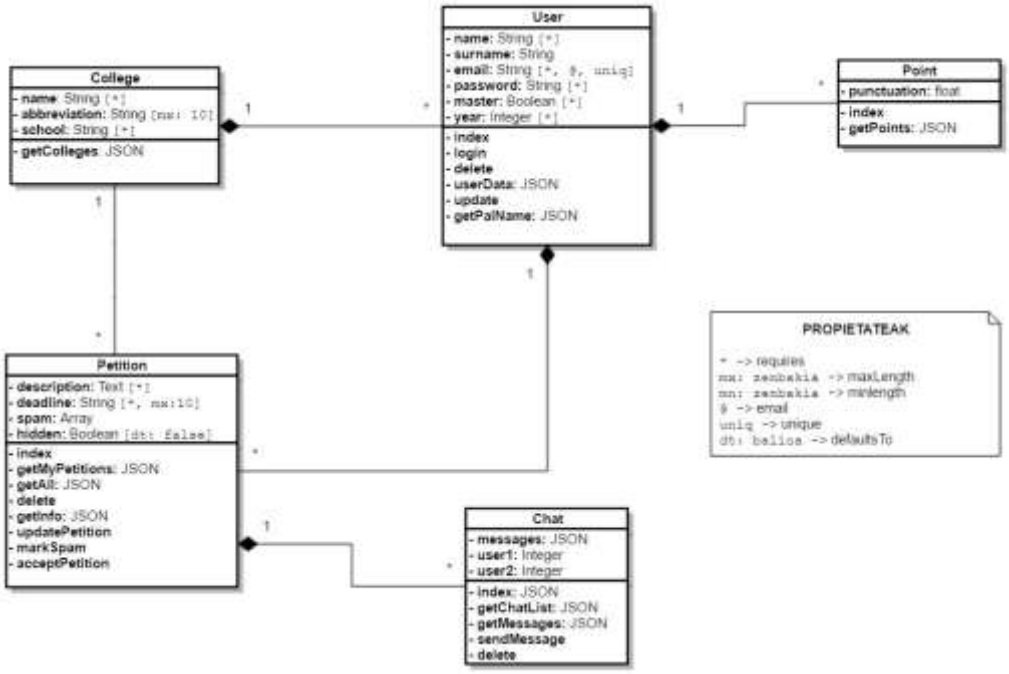
- Laburpena: hitz egiten dagoen beste erabiltzailearen perfila ikustea.
- Aurrebaldintza: sesioa hasita izan eta erabiltzaileak egindako eskaera batean chat bat hasieratuta izan.
- Postbaldintza: datu jakin batzuk ikusiko dira.

Erabiltzailea puntuatu

- Laburpena: laguntza eman edo ez duen erabiltzaile bati 0-10 tarteko puntuazio bat eman.
- Aurrebaldintza: sesioa hasita izan eta erabiltzailearen perfilean sartuta egon.
- Postbaldintza: erabiltzaileari jarritako puntuazioa gehitu egingo zaio.

6.2 Datu ereduak

Atal honetan zerbitzarian gordetzen diren ereduen eta beraien artean egongo diren erlazioak agertzen dira. Ereduen gainean exekutatu daitezkeen funtzioak ere adierazten dira:



6.2.diagrama: datu eredu diagrama

6.2.1 College

Unibertsitateen informazioa gordeko duen taula eta datu egitura izango da. Bertan aurkitu daitezkeen eremuak honakoak dira:

- **name:** unibertsitatearen izena. Honakoa derrigorrezkoa den eremu bat izango da. Adibidez, EHU hemen erregistratu nahi bada, eremu honetan "Euskal Herriko Unibertsitatea" balioa jarri beharko da.
- **abbreviation:** normalean horrela ezagutzen dira unibertsitateak; EHU, LSU, MIT. Izenarekin batera, unibertsitatearen identifikazioa egingo du. Badaude unibertsitate batzuk ez dutela laburdurarik (Cambridge, Oxford) beraz, honako eremua ez da derrigorrezkoa.
- **school:** erregistratu nahi den fakultatea gordetzen duen eremua, informatika adibidez. Unibertsitate batek ez duenez fakultate bakarra izango, honako eremua derrigorrezkoa izango da.

Unibertsitate berri bat sartu nahi bada, erabiltzaileek ez dute eskubide hori izango, lan honen egileak egin beharko du. Beraz, eskaera bat egin beharko da eta ondoren erregistratuko da unibertsitatea.

Erlazioei dagokienez, unibertsitate batean erabiltzaile asko egongo dira eta erabiltzaileek egiten dituzten eskaerak, unibertsitate baten mendeko izango dira. Beraz unibertsitate batek erabiltzaile asko eta eskaera asko izango ditu (One-to-Many erlazioak).

6.2.2 User

Erabiltzaileen informazioa gordeko duen taula eta datu egitura izango da. Bertan aurkitzen diren eremuak honakoak izango dira:

- **name:** aplikazioan erregistratu den erabiltzailearen izena gordeko du. Honakoa derrigorrezkoa izango da.
- **surname:** erabiltzailearen abizena gordeko du. Honako eremua ez da derrigorrezkoa izango, erabiltzaile bakoitzaren esku egongo da datu hau eman nahi duten edo ez.
- **email:** aplikazioan identifikatu daitezen derrigorrezkoa izango den eremua. Gainera, honen bidez pertsona honekin kontaktuan jartzeko aukera egongo da. Sisteman ezin izango dira bi posta elektroniko berdin aurkitu eta hemen gordetzen den datua posta izan behar du, hau da, @ eta luzapen bat (.com, .es, .eus, …) beharrezkoa izango da.
- **password:** kontuaren pasahitza adierazten du. Luzera minimo bat eskatzen da, segurtasuna dela eta (6 karaktere). Gainera, derrigorrezkoa izango da eremu hau.
- **master:** erabiltzaile hori masterreko ikaslea den edo ez adieraziko du. Derrigorrezkoa den datua izango da.
- **year:** erabiltzailea zein kurtsokoa den adierazten duen eremua, derrigorrezkoa ere.

Erabiltzaileak izango dira eskaerak egiten dituzten pertsonak, beraz eskaera askoren jabe ere izango dira (One-to-Many erlazioa). Bestetik, puntuazio bat izango dute ere. Beste erabiltzaileek puntuatuko dute, beraz bertatik balio bat baino gehiago dagozkio (One-to-Many erlazioa). Unibertsitate baten menpe egongo da gainera.

6.2.3 Point

Erabiltzaile baten puntuak adierazten ditu. Aurkitu daitezkeen atributuak honakoak dira:

- punctuation: zenbaki ez oso balio bat izango da eta erabiltzaile bati eman zaion puntuazioa (0-10 tartekoa) izango da.

Erabiltzaile baten menpekoa izango da puntuazio bakoitza.

6.2.4 Petition

Erabiltzaileek egiten dituzten eskaerak gordetzen dituen taula eta datu egitura da:

- description: eskaeraren deskribapena. Deskribapen hauek oso luzeak izan daitezke, beraz ez da mugarik egongo. Derrigorrezko eremu bat izango da.
- deadline: eskaeraren epemuga noiz den adierazten duen eremua. Honakoa derrigorrezkoa den eremu bat izango da eta UUUU-mm-ee egitura jarraitzen duenez, 10 digitu izango dira onartzen den muga (2017-3-5 moduan ere jar daiteke data).
- spam: eskaera hori spam moduan markatu duten erabiltzaileen identifikazioa gordetzen duen eremua.
- hidden: erabiltzaileek eskaera hau ikusten ahalbidetzen duen eremua izango da. Eskaera batean laguntza onartu bada, ez da gehiago ikusiko baina oraindik epemuga ez denez bete, sisteman egon beharko du. Besteek eskaera ikus ez (edo ikustea) dezaten izango da honen betebeharra. Defektuz `false` izango da, hau da, ikusgai egongo da.

Eskaera batek bere menpe chat-ak izango ditu. Erabiltzaile batekin baino gehiagorekin egon daiteke momentuan hitz egiten, beraz bat baino gehiagorekin erlazionatuko da (One-to-Many erlazioa) eskaera bat.

6.2.5 Chat

Bi erabiltzaileen artean bidalitako mezuak gordeko dituen taula eta datu egitura da. Honakoa da aurkitu daitezkeen eremuak:

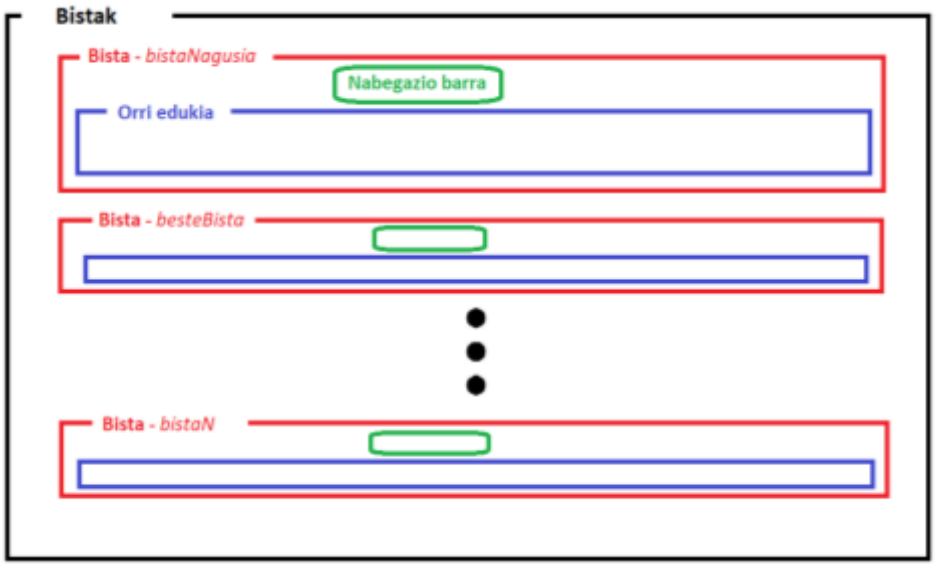
- messages: mezuak gordeko dira eremu honetan. Ondo antolatuta egon dadin, eremua JSON objektuak gordeko ditu.

- **user1:** elkarrizketan parte hartzen duen erabiltzaile baten identifikazio eremua. Eremu honetan elkarrizketa sortu duen pertsonaren identifikazioa egongo da.
- **user2:** elkarrizketan parte hartzen duen beste erabiltzaile baten identifikazio eremua. Hemen norekin hitz egin nahi den pertsonaren (eskaera sortu duen pertsonaren) identifikazioa egongo da.

Eskaera baten menpekoa izango da chat bakoitza.

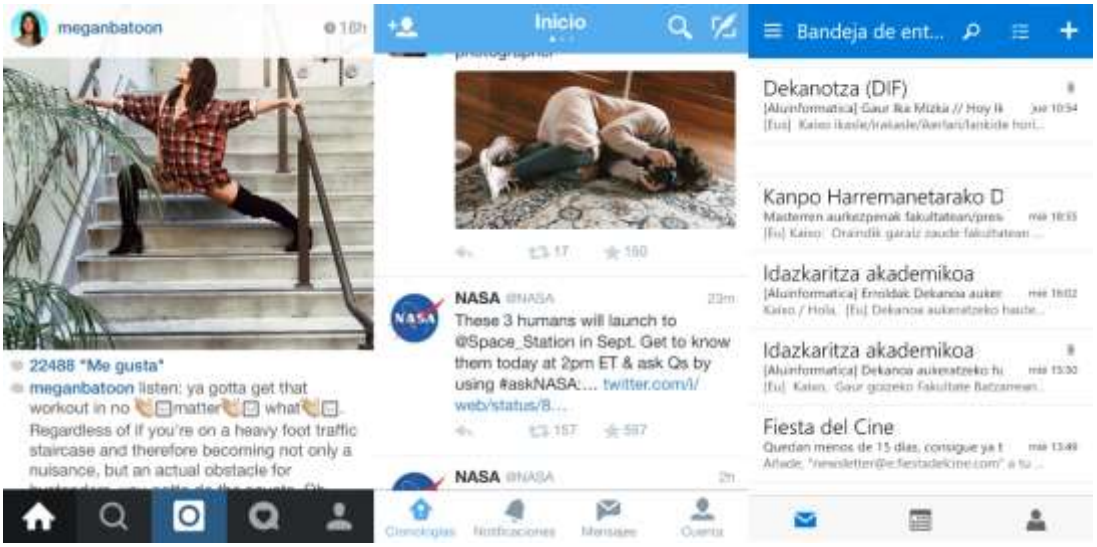
6.3 Bezeroaren pantailen diseinua

Aurreko kapituluan bezeroaren zatia garatzeko Framework7 lan ingurunea erabiliko dela erabaki da. Honi esker bistaz osatutako oinarrizko egitura bat garatu. Bakoitzak bere barnean bere nabigazio barra, edukia eta erremientak izan ditzake. Web aplikaziora eramaten badugu, bista bakoitza HTML fitxategi banaren pareko kontsideratu daiteke. Honen bidez, lan ingurune honetan HTML fitxategi bakarra sortuta, bista desberdinak definitu daitezke eta bere barnean bista desberdinak izan. Baina fitxategi bat baino gehiago sortu daitezke eta beraien artean nabigatzea ere posible da.



6.3.irudia: Framework7 aplikazioaren oinarrizko egitura bat

Egitura honen bidez, bista bakoitza aplikazioko funtzionalitate bat izan daiteke. Instagram, Twitter eta Hotmail aplikazioek egitura hau jarraitzen duten adibide batzuk dira. Hiru hauek nabigazio barra batez, bista desberdinez eta dena orri batean garatuta dago.



6.4.irudia: Instagram, Twitter eta Hotmail aplikazioen bistak

Egitura hau jendearentzako ulererreza, intuitiboa eta ezaguna egiten zaionez, egitura berdina jarraitzen duen aplikazio bat garatzea erabaki da. Ala ere, dena ezin da bisten bidez garatu, eta orri desberdinak sortu beharko dira. Orri desberdinak sortzen direnean HTML fitxategi desberdinak sortuko dira eta batetik bestera nabigatu beharko da.

Orriak estatikoak direnean, adibidez aplikazioaren inguruko informazioa denean, orri hauek web aplikaziotan orri batetik bestera aldatzeko erabiltzen den aingura etiketa (`<a>` etiketa) hemen ere erabili daiteke. Framework7-k orri hauek AJAX bezala kargatuko dituzte eta orrien edukia ikusiko da.

Orriak dinamikoak direnean estrategia hau ez jarraitzea gomendatzen da. Dinamikoa denez, irekitzen denean erakutsiko diren datuak erabiltzaile batetik bestera desberdinak izango dira. Hori dela eta, orri hauek JavaScript bidez irekitzea estrategia hobeagoa da. Horretarako bista baten gainean `router.loadPage()` funtzioa erabili beharko da. Adibidez, `mainView.router.loadPage("chat.hmtl")` funtzioak `mainView`-tik chat orria kargatuko du.

Orri bat ireki edo itxi egiten denean, aplikazioak izango duen jarrera ere kontrolatu daiteke. `index.html` orria berezia izango da. Hau kargatu dela egiaztatu nahi badugu, dispositiboa prest dagoela egiaztatu behar da. Orriak kargatu egin direla egiaztatu nahi bada, 6.5.irudian ikusten den moduan egin beharko da (2. lerroa).

```
// Index hasieratu
$$(document).on('deviceready', function() {
});
// Chat orria hasieratu
myApp.onPageInit('chat', function(page){
});
// Chat orrian atzera
myApp.onPageBack('chat', function(page){
});
```

6.5.irudia: Device Ready, orri bat kargatu eta orritik atzera funtzioak

Aplikazio honetan bi izango dira garatuko diren orriak. Alde batetik hasiera pantaila, chat lista eta perfila bistak izango dituen `index.html` orria. Bestetik, dinamikoki bere datuak aldatuko dituen `chat.html` orria. Bi pertsonen arteko elkarrizketako mezuak erakutsiko ditu, pantaila desberdina sortuz erabiltzaile batetik bestera.

Pop-up leiho batzuk ere sortu egin dira aplikazioan. Hauen inplementazioa berezia da, ez baitute 6.3.irudian definitutako eredua jarraitzen. Alde batetik, ez dute nabigazio barrarik edo erremienta barrarik izango, edukia soilik izango dute. Beraien definizioa gainera bistetatik kanpo egiten da (6.6.irudia). Horrelako leiho bat ireki nahi denean, JavaScript-en `loadModal()` funtzioa erabilita ireki egingo dira.

```
<!-- Profile data + punctuation -->
<div class="popup popup-profile">
</div>

<!-- App views -->
<div class="views tabs toolbar-through">
</div>
```

6.6.irudia: pop-up baten definizioa

Bistekin lan egingo denez, JavaScript fitxategian hauek erazagutu beharko dira. Honakoa aplikazioak erabiltzailea zein bistatik datorren jakin dezan erabiliko da besteak beste. Gainera, bistek propietate bereziak badituzte, hemen ere definitu daitezke propietate hauek.

```
var mainView = myApp.addView('#mainView', {
    dynamicNavbar: true,
});
var chatView = myApp.addView('#chatListView', {
    dynamicNavbar: true,
});
var settingsView = myApp.addView('#settingsView');
```

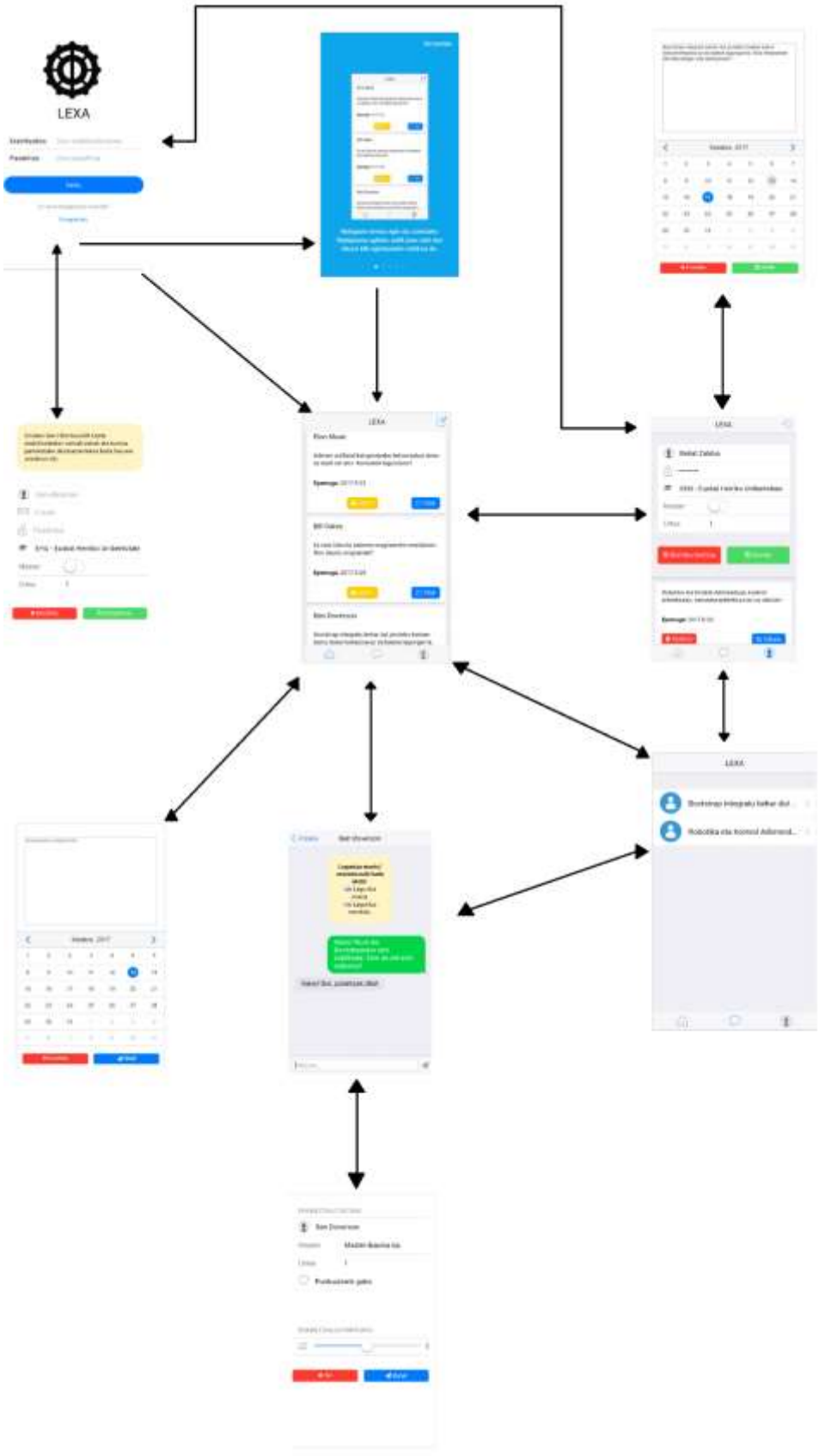
6.7.irudia: bistak erazagutzen

Bistak, orriak eta *pop-up* leihoak sortzeaz gain notifikazioak eta *prompt* leihoak ireki daitezke. Notifikazioak pantailaren goialdean agertzen diren abisuak dira eta egindako aldaketa zein izan den erazagutzen dute. *Prompt* leihoak berriz, pantailaren erdian ateratzen dira eta erabiltzaileek aldaketa bat egin nahi dutenean hau benetan egin nahi dutela egiaztatzeko erabiliko da. Adibidez, webguneetan perfileko datuak aldatu nahi ditudanean benetan aldaketa hori egin nahi dudan leihoak dira. Kasu honetan zerbait sartzeko eskatuko zaio erabiltzaileari.

6.8.irudia: notifikazio adibidea

6.9.irudia: prompt leiho adibidea

Jarraian agertzen den irudian aplikazioko pantaila guztiak agertzen dira eta bata bestetik nabigatzeko dagoen posibilitatea.



6.3.diagrama: aplikazio barruko pantailak eta nabigazioa

6.4 Irisgarritasuna

Irisgarritasuna[16] aplikazio bat erabiltzaile orori sarbidea ahalbidetzean datza, elbarriak alde batera utzi gabe edota kultura desberdineko pertsonak. Kontuan izan beharreko atal bat da kultura, adibidez kolore batek igorri nahi duen mezua kultura batetik bestera desberdina baita.

Existitzen diren elbarritasunen artean begiei eragiten dion bat daltonismoa da. Daltonismoa[17] duten pertsonak ez dira gai koloreak behar bezala bereizteko. Mota desberdinak aurkitzen dira: akromatikoak (txuri-beltzean ikusten dute), monokromatikoak (kolore bakarra ikusten dute, edo horien antzekoak direnak), dikromatikoak (kolore nahasiak ikusten dituzte) eta trikomatiko anomaloa (kolore bat nabaritzeko gaitasuna murriztuta du).

Kulturari dagokionez, aurrerago adibidearekin jarraituz, koloreak esanahi handia du. Europan kolore gorriak kontuz esan nahi du baina Txinan oparotasunaren kolorea da. Hori dela eta, koloreak kontu handiz erabili behar dira.

Aurkitu dezakegun beste arazo bat aplikazioaren internalizazioa izan daiteke. Aplikazio bat hizkuntza desberdinetara itzultzeak denbora asko ekar dezake. Gainera itzuli nahi den hizkuntzara itzultzeko, hau ondo dakien pertsona baten beharra dago.

Aurkeztu egin diren arazo hauei aurre egiteko soluzio sinple baina eraginkorra erabiliko da, ikonoak. Ikonoak zerbait transmititu nahi diguten sinboloak dira eta gehienak internazionalak dira. Adibide argiena ► ikonoa da. Mundu osoan bere esanahia aurrera egin edo martxan jarri esan nahi du. Ala ere, ikono baten esanahia agian ezezaguna izan daiteke pertsona batentzako.

Hau dela eta, ikono eta koloreen arteko konbinazio bat erabili da aplikazioan. Bi baliabide hauen erabilera eginez, gertatu daitezkeen nahastean eragotzi daitezke elementu bakoitzak bere esanahia modu argian igorriko baitu eta gainera aplikazioaren hainbat gauza ez dira itzuli behar.

7

Bezero-zerbitzari arteko konexioa

5.kapituluan azaldu bezala, aplikazioak bezero-zerbitzari moduko arkitektura bat jarraitzen du. Ondoren, MVC egitura bat aplikatuta, zati bakoitzean zer exekutatu egingo den definitu egin da.

Bezero eta zerbitzariaren artean komunikazioa beharrezkoa izango da, bezeroak zerbitzaririk gabe ez baitu funtzionatuko. Kapitulu honetan bien arteko konexioa eta datu trukea nola egiten den azaldu egiten da.

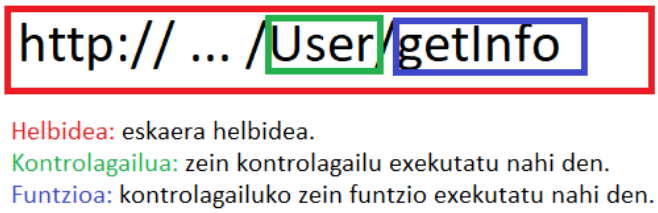
7.1 Zerbitzarira eskaerak

Bezeroek zerbitzariari eskaerak egin beharko dizkio eta zerbitzariak lortutako erantzunak bezeroei jakinarazi. Bien arteko komunikazioa eta datu trukea beharrezkoa izango da beraz. Hau egiten ahalbidetuko digun baliabide bakarra Internet da.

Honen kudeaketa egiteko, zerbitzaria REST (*Representational State Transfer*)[18] moduan garatu egin da. Software arkitektura teknika bat da eta sistema web banatuentzako pentsatuta dago. HTTP (*Hypertext Transfer Protocol*) protokoloan oinarritzen da, beraz egiten diren eskaerak POST eta GET bidez egin daitezke. Arkitektura honetan garatutako zerbitzariak bezeroaren garapenerako abantaila bat da ez baita jakin behar zerbitzaria nola egituratuta dagoen. Ala ere, hainbat gauza jakin behar dira aurretik:

1. Datuen formatua. Zerbitzariak datu mota bat jasoko du eta beste bat itzuliko du (mota berdina edo desberdina izan daiteke). Honakoa aldez aurretik jakitea garrantzitsua da datu formatu okerra bidaltzen bada, zerbitzaria ez da gai izango hau interpretatzeko eta berdina zerbitzariaren erantzuna formatu okerrean badago. Bezeroaren lana da jakitea zein den datuen formatua.
2. Nora egin eskaerak. Datuak ez dira beti helbide berdinera bidaliko, helbide bakoitzak funtzio bat exekutatuko baitu. Horregatik, aldez aurretik nora bidali behar den jakin behar da.
3. Bidaltzen diren datuak egokiak izatea. Zerbitzariak lan egiteko `izena` eta `kodea` datuak behar baditu, hauek bidali beharko dira eta ez `name` eta `code`.

SailsJS lan inguruneak arkitektura honetan oinarritzen diren aplikazioak modu azkar eta errazean garatzea ahalbidetzen du konfiguraziorik egin gabe. Eskaera bat helbide jakin batera egiten da eta helbide horretan zein datu ereduk eta zein funtzio exekutatuko den adierazi beharko da.



7.1.irudia: eskaerak egiteko adibidea

Jarraian zerbitzarira eskaera bat egin nahi bada posible diren kontrolagailuak eta funtzioak adierazten dira. Eskaera bat egiteko lehenengo zerbitzariaren IP helbidea jakin beharko da, ondoren kontrolagailuko balio bat aukeratu eta ondoren zein funtzio exekutatu nahi den, hau da, taulako lehenengo bi zutabeetatik balio bana hartu. Gainerako bi zutabeak eskaera egiteko metodoa eta funtzioaren deskribapen bat agertzen da.

Kontrolagailua	Funtzioa	Eskaera metodoa	Deskribapena
User	Index	POST	Erabiltzaile berri bat sortu.
	Login		Erabiltzailea datu basean dagoen egiaztatu.
	Delete		Erabiltzaile bat borratu
	userData		Erabiltzaile baten datuak itzuli.
	Update		Erabiltzaile baten datuak eguneratu.
College	getColleges	GET	Unibertsitate guztiak itzuli.
Petition	Index	POST	Eskaera berria sortu.
	getMyPetitions		Erabiltzailearen eskaerak itzuli.
	getAll		Unibertsitate bateko eskaerak itzuli.
	Delete		Eskaera bat borratu.
	getInfo		Eskaera baten informazioa itzuli.
	updatePetition		Eskaeraren informazioa eguneratu.
	markSpam		Eskaera spam moduan markatu.
	acceptPetition		Eskaera batean laguntza onartu.
Chat	Index	POST	Chat berri bat sortu.
	getChatList		Erabiltzaile baten chat-ak itzuli.
	getMessages		Chat bateko mezuak itzuli.
	sendMessage		Chat batean mezua bidali.
	Delete		Chat bat borratu.
Point	Index	POST	Puntuazio berria gehitzen dio erabiltzaile bati.
	getPoints		Erabiltzaile baten puntuazioa itzultzen du.

7.1.taula: eskaera helbideak funtzioaren azalpenarekin

`Index` funtzioak ez dira zertan adierazi behar, `/kontrolagailua/` jarrita, zerbitzariak automatikoki ulertzen du exekutatu nahi dena funtzio hori izango dela.

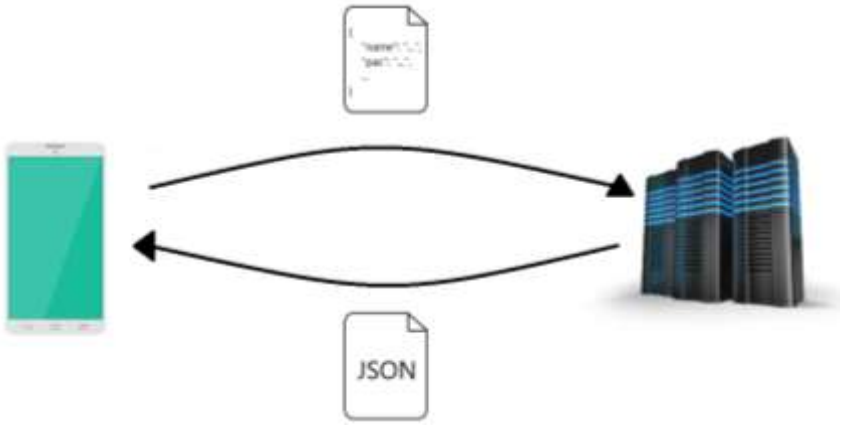
Beraz, zerbitzari eta bezero arteko konexioa egiteko definitu beharreko gauza bakarra datuak bidaltzeko formatua izango da. Datu hauek XML (*Extensible Markup Language*) edo JSON formatuan bidali daitezke.

JSON formatua erabiliko da datu transferentzia egiteko, bezeroaren aldetik eskaerak egiten direnean datuak bidaltzeko eta zerbitzaritik erantzunak bidaltzean. JSON fitxategiak XML fitxategiak baino egokiagoak izango dira AJAX (*Asynchronous JavaScript And XML*) eskaeretan eta erantzunetan datuen tratamendua egiteko. JSON fitxategi bat `Array` moduan tratatu daiteke eta bertatik datuak lortzea erreza izango da. XML fitxategietan berriz, XML DOM erabili behar da hauen tratamendua egiteko, kodea nabarmen luzatuz eta konplikatuz.

Ala ere, zerbitzariak ez du beti JSON fitxategi bat itzuliko. Funtzionalitate batzuk ez dute behar datuz betetako erantzun bat bidaltzea, funtzioa ondo edo gaizki adieraztearekin nahikoa izango da. Hau dela eta, hiru erantzun posible itzuliko ditu zerbitzariak.

1. JSON fitxategiak, erantzunak bere barnean datuak behar dituenean, erabiltzaile baten informazioa lortu nahi denean adibidez.
2. OK mezu bat. Horrelako erantzunak datu basean erregistroak sortu, ezabatu edo eguneratzean itzuliko dira batez ere.

3. Errore mezua. Errore mezu hau mota askotakoa izan daitezke. Honakoa itzultzen bada, errorearen jatorria zein den jakinaraziko da. Errorea zein den adierazteko HTTP protokoloak dituen egoera kodeak erabiliko dira[19]:
    - **500 – Internal Server Error:** zerbitzarian errore bat gertatu dela adierazten du. Exekuzio errore bat izan daiteke, datu mota behar bezala tratatu ez delako adibidez.
    - **404 – Not Found Error:** bilaketa egiten denean sartutako irizpideak betetzen dituen tupla aurkitu ez dela adierazten du. Adibidez, erabiltzaile bat sesioa hasi nahi duenean eta erabiltzaile hori aurkitzen ez bada, honako errorea jaurtiko da.
    - **400 – Bad Request:** eskaera bat modu desegokian egin dela adierazten du. Adibidez, funtzio batek POST bidezko datuak espero dituenean eta GET moduan iritsi direnean.



7.2.irudia: JSON fitxategi bidalketa

Bezero eta zerbitzariaren arteko konexioa zuzena izateko, erantzunaren goiburukoa aldatu beharko da. HTTP goiburukoak HTTP bidez bidalitako datuei buruzko informazioa eskaintzen digute, adibidez bertako datuak zein formatutan dauden.

Kasu honetan, berez ezartzen ez den goiburuko bat gehitu beharko da, *Access-Control-Allow-Origin* goiburukoa[20]. Honen bidez, eskaera egin den helbidera erantzuna bidaltzea onartuko da. Konfigurazio hau segurtasun neurriak dira. Mugikor aplikazio bat garatuko denez, erabiltzaile bakoitzak bere mugikorretik eskaerak egiten dituenean, mugikor horrek izango duen IP helbidea desberdina izango da erabiltzaile batetik bestera. Hori dela eta, atal honetan IP helbide guztiak onartu beharko dira.

7.2 Zerbitzariarekin komunikazioa

Aurreko atalean azaldu den bezala, eskaera guztiak JSON formatuan egingo direla erabaki da. Zerbitzarira datuak bidaltzeko, horrelako objektu bat sortu beharko da baina lehenengo datuak interfazetik lortu beharko dira. Behin hori izanda JSON objektua sortuko da.

```
var postData = {};
postData.email = localStorage.username;
postData.description = $$('.petition').val(); // eskaera deskribapena
postData.deadline = dateFormat(new Date(calendarInline.value[0])); //eskaera data
```

7.3.irudia: eskaera bat egiteko JSON elementua

7.3.irudian ikusten den adibidea jarraitzen badugu, hortik lortuko den JSON objektuak honako egitura izango du:

```
{
    “email”: “erabiltzaile_posta”,
    “description”: “eskaera_deskribapena”,
    “deadline”: “eskaera_epemuga”
}
```

7.1.kodea: postData JSON objektua eskaeretan

Behin objektua prest dagoela, AJAX eskaera bat egin beharko zaio zerbitzariari. Hemen datu transferentzia metodoa (POST edo GET), datu mota, datuak eta eskaera egiteko helbidea zein den adierazi beharko da. Eskaera egin ondoren, zerbitzariak erantzun bat bidaliko du. Erantzun honetan prozedura ondo edo gaizki joan den adieraziko da. Kasu bakoitzean, dagokiona exekutatuko da.

```
$$.ajax({
    method: 'POST',
    data: postData,
    dataType: 'jsonp',
    url: server+'Petition/',
    success: function(){
        myApp.hideIndicator();
        getMyPetitions();
        myApp.closeModal('.popup-petition');
    },
    error: function(){
        myApp.alert('Zerbait gaizki atera da. Mesedez, saiatu berriro.', 'ERROR!');
    }
});
```

7.4.irudia: Ajax eskaera zerbitzarira

Eskaera hauek egin ahal izateko, lan ingurune honek orri nagusian (`index.html` fitxategia kasu honetan) lerro berezi bat gehitzea eskatzen du. *Head* atalean *meta* etiketan *Content-Security-Policy*[21] edo Edukien Segurtasun Polita bete behar da. *Cross-Site-Scripting (XSS)*[22] eta kode injekzioak[23] eragozten dituen eremua izango da, aplikazioaren segurtasun maila bat gehituz. Definitu beharreko atributuen artean zein konexio onartzen diren, onartuko diren fitxategiak non dauden eta erabiltzen diren irudiak non dauden adierazi beharko dira.

Zerbitzariarekin konektatzeko, politika honetan `connect-src` gehitu beharko da. Hemen zerbitzariarekin egiten diren konexio guztiak adierazi beharko lirateke, hau da, zerbitzarira egiten diren eskaera helbide guztiak. Hau dela eta helbide guztiak, edo *, jartzea erabaki da. Segurtasun aldetik honakoa ez da oso segurua, konexio guztiak onartzen baititu.

```
<meta
    http-equiv="Content-Security-Policy"
    content="default-src 'self' 'unsafe-eval' data: gap: https://ssl.gstatic.com;
                style-src 'self' 'unsafe-inline';
                media-src *;
                connect-src *"
>
```

7.5.irudia: Content-Security-Policy atala

8

Aurkitutako zailtasunak eta erroreak

Proiektuan atal batzuk landu egin direnean zailtasunak eta konpondu ezin izan diren errore batzuk aurkitu egin dira. Kapitulu honetan hauek zeintzuk izan diren eta zergatik gertatu diren azaltzen da.

8.1 Chat zerbitzua

Aipatu den bezala, funtzionalitate honek erregistratutako erabiltzaileen artean komunikazioa erraztuko du. Hau ahalbidetzeko mezua bidaltzen duen pertsonak zerbitzarira mezua bidali beharko du eta ondoren zerbitzariak mezu hau dagokion pertsonari bidaliko dio.

Denbora errealeko mezularitza aplikazioek zerbitzariarekin duten konexioa ez dute ixten, hau da, atzetik socket-ak erabiltzen dira. Socket-en bidez zerbitzariarekin egiten den konexioa

ez da inoiz eteten espreski horrela eskatzen ez bada. Konexioa dagoen bitartean datuak jaso edo bidali egingo dira egoeretan oinarritutako programazio bat jarraituz.

Framework7 eta SailsJS JavaScript erabiltzen dute. Framework7 lan ingurunean socket bat sortu nahi bada, Socket.io[24] deitzen den JavaScript liburutegi baten beharra dago. Honekin socket-a sortu, datuak jaso eta mezuak bidaliko dira. SailsJS socket-en erabilera ere onartzen du. Hauen konfigurazioa egin daiteke lan ingurunean bertan.

SailsJS eta denbora errealeko mezularitza aplikazio bat aurkitu egin da GitHub-en, sailsChat[25] izena duena, baina hemen jarraitzen den programazioa jarraituta, ez da lortu socket bidez konexioa.

Aplikazioaren bi atalak socket bidez elkar komunikatzea lortu ez denez (eta garapen denboran desbideraketa sortzen zegoenez) AJAX erabiltzea erabaki da soluzio moduan. AJAX zerbitzariarekin konektatu egingo da baina erantzuna jasotzen duenean konexio hau itxi egingo da. Dagoeneko modu honetako eskaerak egin direnez, estrategia hau jarraitzea erabaki da.

Honek aplikazioan eragin txar bat du. Garapena horrela egitean chat-a ez da denbora errealeko mezularitza zerbitzu bat izango, mezularitza zerbitzu bat baizik. Mezuak denbora tarte batean (1000ms definitu da) begiratzen egon beharko du bezeroak, zerbitzariarekin konexioa behin eta berriz irekitzen eta itxiz. Honek Internet eta bateria kontsumo handia ekar dezake.

8.2 Zerbitzariarekin konexioa galdu

7.2.atalean azaltzen den bezala, zerbitzari eta bezeroa konektatzeko Framework7 lan inguruneak *Content Security Policy* etiketa gehitzea eskatzen du. Zehaztu beharreko atributuen artean `connect-src` atributua dago, aplikazioak onartuko dituen konexioak zehazten dituela.

Interneteko dokumentazioan begiratuta IP-ak jarri behar direla aipatzen da eta * ikurra (helbide guztiak adierazten duela) ez dela onartzen. Beraz, zerbitzarira egiten den eskaera bakoitzeko zerbitzariko helbide bat gehitu beharko litzateke eremu horretan.

Bat batean, aplikazioak funtzionatzeari utzi zion. Segurtasun kopiak berreskuratu eta gero, bai disko gogorrean zeudenak, bai Git-en zeudenak, momentura arte jarraitutako egitura jada ez zuen funtzionatzen, hau da, IP helbideak jartzea ez zen soluzioa.

Ez da aurkitu hau azaltzen duen arrazoi bat, baina zerbitzaria eta bezeroa ez ziren konektatzen. Ala ere, honakoa arrazoi hauengatik gertatu zitekeela pentsatu da.

- Zerbitzariaren arazoa da. NodeJS bertsioa aktualizatzean SailsJS-rekin gatazka sortzen zuen zerbait gertatu zitekeen.

- PhoneGap aktualizatu izana. Aktualizatu izanak, *Content Security Policy*-ren aldaketa ekar dezake.
- Ordenagailuaren funtzionamendu okerra. Aktualizazioak direla eta aplikazioak ez funtzionatzea.

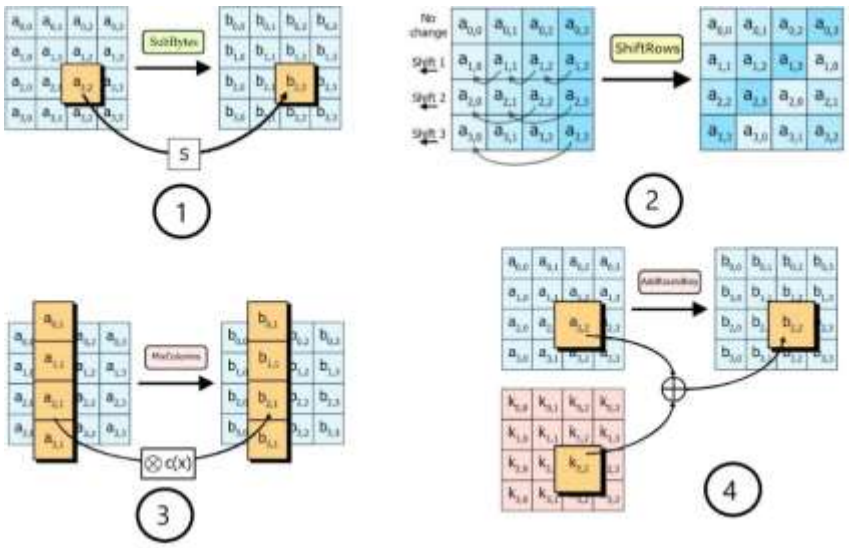
Nahiz eta honako arazoak pentsatu izana, ez da kasu hauetako ezta bat eman. Azkenean `connection-src` atalean * jarri da, hasiera batean honakoa onartzen ez zuela pentsatuz, eta bezero-zerbitzari konexioa berriz funtzionatzen zuen.

Arazoa zein izan den oraindik ez da identifikatu. Aplikazioren batean (PhoneGap edo NodeJS) onartu gabeko aktualizazio edo aldaketa bat gertatu dela ondorioztatzen da, zaharragoak ziren bertsioak martxan jartzean ez zutelako funtzionatzen eta garatu zirenean funtzionatzen zuten.

8.3 Segurtasuna aplikazioan

Bi atalak garatu egin direnean, hainbat segurtasun maila hartu egin dira: korreoa bidaltzea eta *Content Security* Policy gehitzea baina datu pertsonalekin lan egiten denez, segurtasun maila bat gehiago gehitzea oso beharrezkoa ikusi da.

Dauden enkriptazio algoritmoetatik AES (*Advanced Encryptation Standard*)[26] da seguruena, sortu zenetik eraso bakarra izan da erregistratuta arrakastatsu bezala. *Rijndael* izenarekin ere ezagutzen da baina ez da gauza bera; *Rijndael* algoritmoak erabiltzen dituen bloke-tamainak eta gako-luzera AES erabiltzen dituenak baino handiagoak dira. AES-ek bloke tamaina finkoa du, 128 bit eta gakoaren tamaina 128, 192 edo 256 bitekoa izan daiteke. AES egoera matrize (*state*) bat erabiltzen du bere eragiketa guztiak egiteko. Matrize hau 4x4 tamainakoa da eta batzuetan zutabeak ugariagoak dira erabiltzen den *Rijndael* bertsioagatik.



8.1.irudia: AES enkriptazio pausoak

Algoritmo hau inplementatu nahi denez, JavaScript-en hau egiten ahalbidetzen duen liburutegi edo kodea bilatu da Interneten. CryptoJS liburutegia[27] aurkitu egin da, AES algoritmoa emateaz gain beste enkriptazio mota desberdinak dituena. Bezeroaren aldean liburutegi hau jartzeko, deskargatu egin behar izan. Ondoren `<script>` etiketa jartzen da `index.html` fitxategian. Zerbitzarian NodeJS modulu bezala instalatu behar da `npm install crypto-js` komandoa erabiliz. Behin hau eginda, hasierako konfigurazioa eginda dago.

Bezeroaren aldean enkriptatuko diren datuak zerbitzarira bidaliko diren datuak izango dira. 7. Kapituluan azaltzen den bezala, zerbitzarira bidaliko diren datuak JSON formatuan egongo dira, beraz objektu hau enkriptatu egingo da. Ondoren zerbitzarian, desenkriptazioa egitean berriro ere JSON moduan tratatu daitezke datu hauek.

Estrategia hau ezin izan da aurrera eraman enkriptazio liburutegia dela eta. JSON objektua enkriptatzen saiatzean, errore mezu bat jaurtitzen da esanez enkriptatzen dagoen datuak luzeegiak direla. Beste aukera bat, JSON objektuak dituen elementu bakoitzeko, bertako datuak enkriptatzea izango litzateke, baina hau errendimendu aldetik, ez da batere egokia. Bidali nahi den datu bakoitza enkriptatzeak denbora gehiago hartzen du objektu oso bat enkriptatzea baino. Gainera, nolabait pertsona batek datuak bidean antzematen baditu, nahiz eta datuak enkriptatuta egon, bertan zer bidaltzen den jakingo du, elementuaren identifikazioa ez baitzan enkriptatuta egongo.

JSON objektua enkriptatu ezin izan denez, segurtasun maila ez da inplementatu. Ala ere, honakoa garatu beharreko puntu bat da, oso garrantzitsua baita. Lan hau garatzeko dagoen denbora dela eta, ezin da behar bezalako denbora inbertitu hain garrantzitsua den puntu bati. Proiektuarekin aurrera jarraitzen bada, honakoa lantzeko egongo den puntu bat izango da.

9

Hobekuntzak

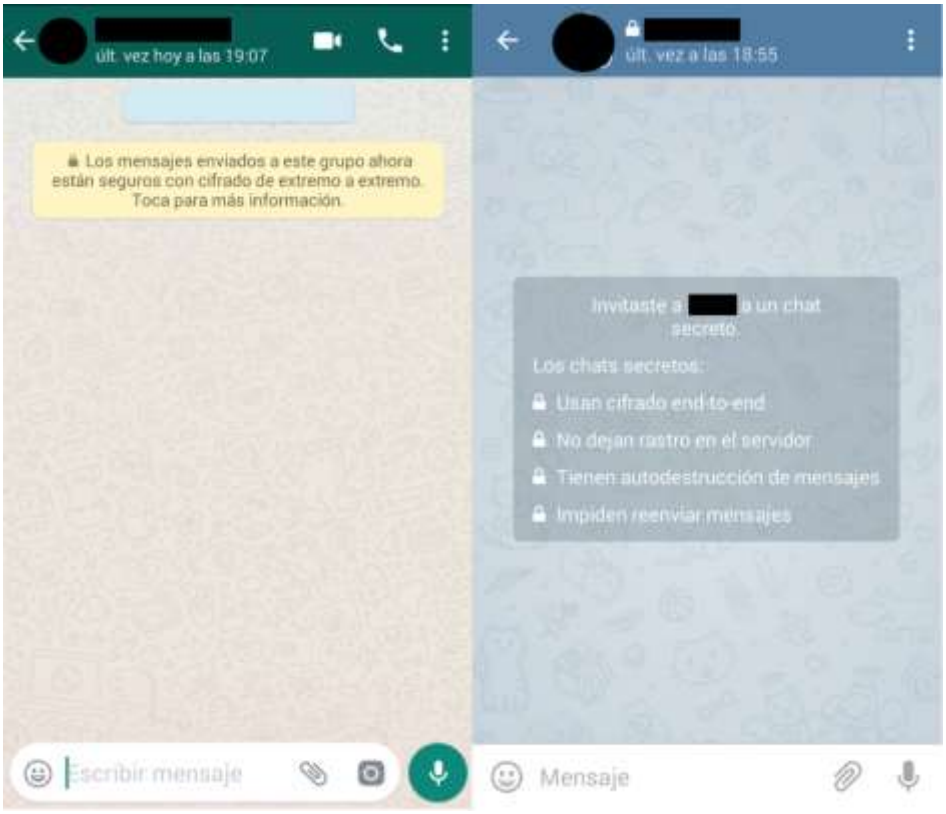
Aplikazioa funtzionalitate batzuekin gelditzen da baina hauek zabaltzeko eta hobetzeko aukera badago. Gehitu daitezkeen funtzionalitate berriak aplikazioa erabili dezaketen pertsonei galdeketa bat egin ondoren jasotako erantzunetatik ateratzen dira.

9.1 Chat hobekuntza eta funtzionalitate berriak

Garapen prozesuan aurkitutako errore handienetako bat denbora errealeko mezularitza zerbitzu bezala garatzea lortu ez izana izan da, hau da, socket-en inplementazioa (8.1. atala). Honakoa lortzen bada, mezuak jasotzeko arazoa ez litzateke gertatuko. Gainera, erabiltzaileak konektatuta dauden, deskonektatu diren edota mezuak irakurri diren.

Hemen ere segurtasun maila bat gehitzea interesgarria izango litzateke, P2PE (*Point-to-point encryption*)[28] edota E2EE (*End-to-End* Encryption)[29] deritzen enkriptazio mota bat

garatzea interesgarria da. Jarrera hau dagoeneko mezularitza aplikazio ezagunenak (WhatsApp eta Telegram) barneratuta dute. Modu honetan, mezuak guztiz pertsonalak izango dira bi erabiltzaileen artean, eta zerbitzarian gordetzen dena, ezingo da ulertu erasoren bat badago.



9.1.irudia: WhatsApp eta Telegram enkriptazio mezuak

Bestetik, erabiltzaileek eskatu egin duten beste hobekuntza bat chat-ean fitxategiak, irudiak eta audioak bidaltzeko aukera izan da. Honen bidez mezularitza zerbitzu hau askoz ere funtzionalagoa izatera pasatuko litzateke.

Denbora dela eta aurrera eraman ezin izan ziren *push* motatako notifikazioak ere garatzea puntu gehigarri bat izango litzateke aplikazioarentzako. Hauen bidez aplikazioak mezu berri bat dagoenean abisua emango luke eta ez zan zertan aplikazioan sartu behar mezu berriak dauden egiaztatzeko.

9.2.irudia: push notifikazioen adibidea

9.2 Perfilak hobetu

Momentu honetan perfilak testu sinplez osatuta daude. Perfila ikusten duen erabiltzaileak datu gutxi batzuk jasotzen dituzte (izen-abizenak, zein urtekoa den, masterrekoa den eta puntuazioa).

Pertsona batekin geratu nahi bada, pertsona hau fisikoki nolakoa den jakitea interesgarria iruditu zaie. Hau perfiletan irudiak onartuz egin daiteke. Edozein aplikazio motetan, perfilak gaur egun argazkiekin datoz pertsonaren identifikazioa egiteko, beraz hau gehitzea ere oso interesgarria izango litzateke.

9.3 Eskaera mota desberdinak sortu

Garapen honen ondoren, eskaerak ikasketa zalantzak zein diren eta honekin laguntzeko norbait bilatzea du helburu, hau da, laguntza akademikoa bilatzea.

Laguntza akademikoaz gain beste motatako eskaerak egitea ere eskatu egin dute galdetutako pertsonek. Adibidez, ikasle bat Donostian bizi dena Beasainera praktikak egitera joan behar badu, baina kotxea ez duena, eta beste bat leku berdinera joan behar duena baina kotxea duena, hitz egin dezakete kotxea partekatzeko.

Azaldutako adibidea dagoeneko existitzen diren aplikazioen eraginez eskatu da (Uber, BlaBlaCar, …) baina ez da kotxeekin soilik. Fakultateko eguna antolatzeko mezuak, RITSI-ra joateko abisuak edota ekintza berezietan parte hartu nahi duen jendea bilatzeko izan daitezke.

Honen aurrean, eskaerak kategorizatzeko beharra sortzen da, eta eskaera guztiak ez dute zertan fakultate mailan geratu behar, unibertsitate mailan agertu daitezke. Modu honetan,

unibertsitatearekin erlazionatuta sortu daitezkeen behar guztiak, aplikazio bakar batekin konpondu daiteke.

Bibliografia

[1] *Ley Orgánica de Protección de Datos (LOPD).*

Webgunea

http://www.lopd-proteccion-datos.com/ley-proteccion-datos.php

[2] *Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico (LSSI)*

Espainiako gobernuko webgunea

http://www.lssi.gob.es/paginas/Index.aspx

[3] Gantt, Linking Tasks.

Webgunea

http://www.gantt.com/creating-gantt-charts.htm#linking-tasks

[4] Scrum (desarrollo de software)

Wikipedia

https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

[5] Native vs. Hybrid

Webgunea

http://www.raona.com/es/Solutions/Template/163/App-nativa-web-o-híbrida-

[6] Ionic

Webgune ofiziala

https://ionicframework.com/

[7] Framework7

Webgune ofiziala

http://framework7.io/

[8] React Native

Webgune ofiziala

https://facebook.github.io/react-native/

[9] CakePHP

Webgune ofiziala

https://cakephp.org/

[10] Django

Webgune ofiziala

https://www.djangoproject.com/

[11] SailsJS

Webgune ofiziala

http://sailsjs.com/

[12] PhoneGap

Webgune ofiziala

https://phonegap.com/

[13] Postman

Webgune ofiziala

https://www.getpostman.com/

[14] Thin Client

Wikipedia

https://en.wikipedia.org/wiki/Thin_client

[15] MVC – Model-View-Controller

Wikipedia

https://en.wikipedia.org/wiki/Model-view-controller

[16] Web Irisgarritasun

Wikipedia

https://eu.wikipedia.org/wiki/Web_irisgarritasun

[17] Daltonismo

Wikipedia

https://eu.wikipedia.org/wiki/Daltonismo

[18] REST – Representational State Transfer

Wikipedia

https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

[19] HTTP Status Codes

Wikipedia

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

[20] Control de Acceso HTTP (CORS)

Mozilla Firefox Dokumentazioa

https://developer.mozilla.org/es/docs/Web/HTTP/Access_control_CORS#Access-Control-Allow-Origin

[21] Content Security Policy (CSP)

Webgunea

https://content-security-policy.com/

[22] Cross-Site Scripting (XSS)

Wikipedia

https://eu.wikipedia.org/wiki/Cross-site_scripting

[23] Code Injection

Wikipedia

https://en.wikipedia.org/wiki/Code_injection

[24] Socket IO

Webgunea

https://socket.io/

[25] SailsChat – Scott Gress

GitHub proiektua

https://github.com/sgress454/sailsChat

[26] AES – Advanced Encryption Standard

Wikipedia

https://eu.wikipedia.org/wiki/Advanced_Encryption_Standard

[27] CryptoJS

Google Code

https://code.google.com/archive/p/crypto-js/

[28] Point to Point Encryption (P2PE)

Wikipedia

https://en.wikipedia.org/wiki/Point_to_Point_Encryption

[29] End-to-End Encryption (E2EE)

Wikipedia

https://en.wikipedia.org/wiki/Point_to_Point_Encryption

[30] SailsJS Email – Balderdash

GitHub proiektua

https://github.com/balderdashy/sails-hook-email

A eranskina

Bileren txostena

Eranskin honetan Imanol Usandizaga Lombana, Gradu Amaierako Proiektuaren tutorea, eta Beñat Zaldua Del Olmo, proiektuaren garatzailea, egin dituzten bileren aktak jasotzen dira.

2017/03/29-ko bilera

Bileraren informazio orokorra:

- Bilerara deitutako pertsonak: Imanol Usandizaga eta Beñat Zaldua.
- Bilera data: 2017/03/29 – 11:00
- Lekua: Imanolen bulegoan.
- Jorratuko diren gaiak:
  1. Proiektuaren nondik norakoak.

2. Garatutakoa aztertzea.
3. Maketa erakutsi.
4. Galderak eta gomendioak.

Bilera

Imanolek eskatuta, bilera atzeratu egin da. Biltzeko ordu berria 12:30.

Bilerara deitutako pertsona guztiak elkartu egin dira, beraz bilerari hasiera eman zaio (12:30)

1. Proiektuaren nondik norakoak

- Aplikazioa bi zatitan banatuko dela aipatu da (*thin client* bat). Zerbitzaria *REST API* bat izango dela adierazi da eta bertara egiten diren eskaera guztiak AJAX bidez egingo direla (Chat-a kenduta).
- Bezeroaren aldea garatzeko erabiliko dituen teknologiak aurkeztu egin dira. Bezeroarentzako Framework7 erabiliko da, HTML, CSS eta JS-en oinarritzen dela eta zerbitzaria garatzeko SailsJS erabiliko da, MySQL datu basearekin.
- Chat-a garatzeko Socketi.io erabiliko da. XMPP softwarea ezagutzen duen galdetu du Beñatek. Imanolek ezetz esatean, hasierako estrategiarekin jarraitzea erabaki da.
- Erabiliko diren erremienta gehigarriak aurkeztu eta zertarako izango diren aipatu da: MySQL Workbench, Postman, Sublime, Chrome eta PhoneGap.

2. Garatutakoa aztertzea

- Hainbat funtzionalitate funtzionalak dira (Login, Logout, Erregistroa). Kontua borratzearen funtzionalitatea oraindik garapen prozesuan dago.
- Garapena azkarragoa izateko, hasieran aplikazioaren maketa bat egin da, nola gelditukо den ikusteaz gain, funtzionalitateak soilik programatzeko hemendik aurrera.

3. Maketa erakutsi

- Aplikazioa martxan jarri da. Imanolek probak egin ditu Beñaten gainbegiratzearekin.
- Testua erabiltzearen ordez irudiak erabiltzearen arrazoia azaldu da (irisgarritasuna).
- Imanolek erregistro, login eta logout funtzionalitateak probatu egin ditu.
- Gustuko izan du aplikazioaren maketa. Ez du arazo handirik izan aplikazioarekin lan egiteko.

4. Galderak eta gomendioak

- *Content Security Policy*-ri buruz galdetu egin zaio Imanoli. Segurtasunarentzako den gune bat dela adierazi du baina zehatz mehatz zer jarri behar den ez daki. Honen inguruan gehiago irakurriko dela aipatu da.
- Tutorearen rola zein den galdetu da.
- Memoriarekin hastea gomendatu da.
- Kontsultatzen eta garatzen den guztia nonbaiten apuntatzea gomendatu da (dagoeneko egiten da).

Bileran hartutako erabakiak

1. Teknologiak eta erremientak egokiak iruditu zaizkio, hauekin jarraituko da lanean.
2. REST API moduan garatzea ondo iruditu zaio, estrategia hau aplikatuko da beraz.
3. Maketa egokia iruditu zaio. Hasiera batean ez dira aldaketarik egingo.
4. Memoriarekin hastea erabaki da.
5. Garapenarekin jarraitu.

Erabaki hauek hartuta eta nahi izan den guztia hitz eginda, bilerari amaiera eman zaio (13:30).

2017/05/03-ko bilera

Bileraren informazio orokorra:

- Bilerara deitutako pertsonak: Imanol Usandizaga eta Beñat Zaldua.
- Bilera data: 2017/05/03 – 12:00
- Lekua: Imanolen bulegoan.
- Jorratuko diren gaiak:
  1. Proiektuaren egoera.
  2. Aplikazioa probatu.
  3. Memoriaren inguruko galderak.
  4. Galderak eta gomendioak.

Bilera

Bilerara deitutako pertsona guztiak Imanolen bulegoan elkartu egin dira. Bilerara hasiera eman zaio (12:00).

1. Proiektuaren egoera

- Funtzionalitate guztiak garatuta daude. Hauetan gauza batzuk txukuntzea falta da baina dagoeneko guztiz funtzionala da.

- Chat-a arazoak eman dituela azaldu da. Socket bidez egiten saiatu eta gero, azkenean AJAX bidez egin da. Funtzionatzen duen bitartean, horrela uztea aipatu da, baina denbora badago hau hobetu nahi da.
- Segurtasun mailarik ez dagoela aipatu da. Denbora badago, hau inplementatzeko denbora badago, inplementazioa egiten saiatuko da. Inplementatzekotan, AES 256 erabili nahi da.

2. Aplikazioa probatu
   - Imanolek aplikazioa probatu egin du mugikorrean eta aldi berean ordenagailuan jarri da.
   - Spam moduan markatzean ez zaio argi gelditu. Berriz egitean, argi ikusi du notifikazioa esanez spam moduan markatu egin dela.
   - Chat-ean bidaltzearen botoiarekin arazoak (mugikorraren teklatuaren gauza izan daiteke).
   - Ordenagailuan, chat-ak errorea eman du (mezua behin baino gehiagotan bidali da). Hau batzuetan gertatzen dela aipatu da. Egoera aztertu eta proba gehiago egingo direla aipatu da.
   - Datuak eguneratzean/erregistroan, urtea aukeratzeko ikonoa ez da oso argia. Hau testuarekin edo aldatzea erabaki da.

3. Memoriaren inguruko galderak
   - Letraren tamaina eta iturriaren inguruan galdetu da. Eredua hartzeko esan da.
   - Beste urtetako memoria batzuk begiratu egin dira ikusteko nolakoak diren.

4. Galderak eta gomendioak
   - Aurkeztutako lanarekin pozik daude.
   - Bilerak beharrezkoak izan direnean bakarrik eskatu egin direla komentatu da. Ez zegoenez gauza handirik erakusteko eta proiektuaren kontrola lanaren egilearen betebeharra denez, bere esku gelditzen da hau. Ados daude bi aldeak ideia honekin.

<u>Bileran hartutako erabakiak</u>

1. Chat-a socket bidez inplementatzen saiatu. Horretarako emango zaion denbora 2 egunekoa izango da. Ikusten bada ez dela lortzen zerbitzaria eta bezeroa konektatzea honen bidez, alde batera utziko da.
2. Segurtasun maila bat inplementatu. Honi dedikatuko zaion denbora 1 egun izango da. Ikusten bada ez dela lortzen, alde batera utziko da.
3. Diseinuan, erregistroan/datuak eguneratzeko pantailan, urtea aukeratzeko ikonoa aldatu egingo da, testu edo beste ikono esanguratsu batengatik.
4. Chat-eko errorea aztertu eta aurkitzen bada konpondu.

Erabaki hauek hartuta, bilerari amaiera eman zaio (13:00).

B eranskina

Testen txostena

Atal honetan, garatutako funtzionalitateak probatzean ateratzen den testen txostenak jasotzen dira. Beraz, egiten diren testen deskribapen bat, espero den erantzuna eta jasotako erantzunak agertuko dira. Beharrezkoak direnean, test hauek oharrekin datoz.

Hemen agertzen diren testen erantzunak ez dira proba bakarra egin ondoren jasotzen den erantzuna izango, behin baino gehiagotan (5 aldiz gutxienez) probatu egin dira eta honakoa dokumentatu egin da. Erroreren bat gertatzen bada, funtzionalitate horren testak gelditu egingo dira, ondo funtzionatzen ez duela dokumentatuko da eta gainerako funtzionalitateak probatuko dira. Ondoren, funtzionatzen ez duen hau konpondu egingo da.

03/21-03/24 testak

03/21 eta 03/24 daten artean garatutako funtzionalitateak probatu egingo diren gauzak honakoak dira:

1. Zerbitzariarekin konektatzen da.
2. Zerbitzariak erantzuna bidaltzen du.

Behin hau probatuta, garatutako funtzionalitate batzuk probatuko dira, hauek zehatz mehatz:

1. Erregistroa.

Erantzunen taula

Deskribapena	Espero den erantzuna	Lortutako erantzuna
Zerbitzaria martxan, mezua bidali.	Zerbitzaritik OK mezua jaso.	OK mezua jaso da zerbitzaritik.
Zerbitzaria ez dago martxan.	Zerbitzaria prest ez dagoela jaso.	Zerbitzaria ez da aurkitu.
Erregistro datuak ondo bidali.	Datu basean erregistro berri bat.	Datu basean erregistro berri bat dago.
Erregistro datuak gaizki, posta errepikatua.	Errorea zerbitzarian. Bat existitzen da.	Errorea zerbitzarian, datu horiekin erregistro bat dago.

B.1.taula: test erantzun taula

Lortu egin diren erantzunak espero egin diren erantzunak izan dira, beraz garapena ondo egin da. Ez dira beste probarik egin, ez baita ikusten beharra. Hau dela eta, *sprint* hau bukatutzat emango da eta hurrengoa garatzen hasiko da.

03/27-03/31 testak

03/27 eta 03/31 daten artean garatutako funtzionalitateak probatu egin dira. Garatutako diren funtzionalitateak eta probatu egingo direnak honakoak dira:

1. Login, sesio hastea.
2. Logout, sesioa itxi.
3. Kontua borratu.

4. Profila editatu.
5. Tutoriala gehitu.

Proba batzuetan datuak nahita gaizki sartuko dira errore mezuak jaurtitzen diren probatzeko.

Erantzunen taula

Deskribapena	Espero den erantzuna	Lortutako erantzuna	Oharrak
Login, datu okerrekin.	Errore mezua ikusi.	Errore mezu bat jaurti da bezeroan.	
Login, lehenengo aldiz, datu egokiekin.	Login itxi, hasiera pantaila ikusi.	Login itxi da. Hasiera pantaila ikusten da, sesioa hasita.	
Login bigarren aldiz, logout ez da egin.	Modu automatikoan login.	Login itxi da eta hasiera pantailan, sesioa hasita modu automatikoan.	Logout ez da egin behar, aplikazioa ixtean eta berriz irekitzean gertatzen da hau.
Logout	Login pantailara joan.	Sesioa itxi. Login pantailara joan da.	
Kontua borratu	Kontua borratu.	Kontua borratu da, login pantailara bidali erabiltzailea.	Pasahitza sartu behar da borratzeko.
Profila editatu	Datuak eguneratuko dira, notifikazio bat abisatzen.	Datuak eguneratuko dira, notifikazio bat erabiltzaileari abisatzeko.	Pasahitza sartu behar da datuak editatzeko.
Tutoriala gehitu (*)	Tutoriala ikusiko da.	Tutoriala ikusiko da.	Aplikazioa exekutatzen den lehenengo aldian edo datuak borratzean ikusiko da soilik.

B.2.taula: test erantzun taula

(*) : honako funtzionalitateak aurrerago aldaketa bat izango dute. Funtzionalitate aldetik ez da aldaketa bat izango, diseinu aldaketa bat izango da.

Lortu diren erantzunak espero ziren erantzunak izan dira. Hau ikusita, hurrengo funtzionalitateak garatuko dira, *sprint* honi bukaera emanez.

04/03-04/07 testak

04/03 eta 04/07 daten artean garatutako funtzionalitateak probatu egingo dira. Probatu egingo diren funtzionalitate berriak honakoak dira:

1. Fakultate berri bat gehitu zerbitzarian.

Eta aurreko puntu batean garatutako hainbat funtzionalitate berreskuratu egingo dira bertan aldaketak egin baitira:

1. Erregistroa, orain fakultatea aukeratu daiteke.
2. Perfila editatu, orain fakultatea aukeratu daiteke.

Erantzunen taula

Deskribapena	Espero den erantzuna	Lortutako erantzuna
Gehitu fakultatea aplikazioan.	Datu basean erregistro berri bat gehitu.	Datu basean erregistro berri bat sortu da.
Erregistroan fakultate bat aukeratu.	Erregistro berri bat sortu.	Erregistro berri bat sortu, dagokion unibertsitatearen erreferentziarekin.
Perfila editatzean, fakultatea aldatu.	Datuak aldatu, unibertsitate berrira erreferentzia jarriz.	Erregistroan, erabiltzailearen unibertsitatearen erreferentzia aldatu da.

B.3.taula: test erantzun taula

Erantzunak espero izan direnak izan dira, beraz garapenarekin jarraitu egingo da, *sprit* honi bukaera emanez.

04/10-04/14 testak

04/10 eta 04/14 daten artean garatutako funtzionalitateak probatu egin dira. Probatu egingo diren funtzionalitate hauek honakoak dira:

1. Eskaera berri bat egin.
2. Eskaera bat borratu.
3. Eskaera bat editatu.
4. Eskaera pertsonalak kontsultatu.
5. Eskaera guztiak ikusi

Erantzunen taula

Deskribapena	Espero den erantzuna	Lortutako erantzuna	Oharrak
Eskaera berri bat gehitu, datu egokiekin.	Erregistro berri bat datu basean.	Erregistro berri bat gehitu datu basean, notifikazioa ikusi.	
Eskaera berri bat egin, deskribapena gabe.	Errore mezua.	Errore mezua, deskribapena eskatzen da.	
Eskaera berri bat, data okerrarekin.	Errore mezua.	Errore mezua, data okerra dela esanez.	Data gaurko eguna baino beranduago izan behar du.
Eskaera bat borratu.	Eskaera borratu eta notifikazioa ikusi.	Eskaera borratu, notifikazioa borratu, lista eguneratu.	
Eskaera editatu, datu egokiekin.	Eskaera eguneratzen da.	Eskaera eguneratu egin da	
Eskaera editatu, deskribapena borratu.	Errore mezua.	Errore mezua. Deskribapena behar du.	
Eskaera editatu, data okerra jarriz.	Errore mezua.	Errore mezua. Data okerra.	Data gaurko eguna baino beranduago izan behar du.
Eskaera pertsonalak ikusi.	Eskaera lista ikusi.	Eskaera lista ikusiko da.	Eskaera bat ezabatzean edo eguneratzean, hemengo datuak eguneratzen dira.
Eskaera guztiak ikusi.	Eskaera guztiak ikusiko dira.	Eskaera guztiak ikusi.	

B.4.taula: test erantzun taula

Probak guztiek espero izan diren erantzunak itzuli dituzte, beraz fase hau bukatutzat emango da. Hurrengo faseari hasiera emango zaio aplikazioari funtzionalitate gehiago gehituz.

04/14-04/21 testak

04/14 eta 04/21 daten artean garatutako funtzionalitateak probatu egingo dira. Denbora tarte honetan garatutako funtzionalitateak honakoak izan dira:

1. Chat zerbitzua.

Erantzunen taula

Deskribapena	Espero den erantzuna	Lortutako erantzuna	Oharrak
Chat berri bat sortu.	Chat erregistro berri bat sortu, listan chat berria ikusi, chat pantaila ireki.	Chat erregistro berri bat sortu da, listan chat berri bat ikusi eta chat pantaila ireki da.	Botoi berdina sakatzean, chat hori ireki egingo da.
Mezua bidali.	Pantailan mezu berria idatzi.	Pantailan mezu berria bidaltzen da.	
Mezuak jaso.	Pantailan mezu berriak pantailaratu.	Mezu berria badago, pantailan idazten da.	Atzerapen txiki bat gertatu daiteke bidaltzen denetik jasotzen den arte.
Chat irekitzean mezuak ikusi.	Aurretik bidali diren mezuak kargatu.	Aurretik bidalitako mezuak pantailan ikusten dira.	Aurretik mezurik egon badira soilik pantailaratzen dira, bestela elkarrizketa hutsa da.

B.5.taula: test erantzun taula

Funtzionalitate honek aurrerago aldaketa batzuk jasango ditu, beraz honen gainean proba gehiago egingo dira. Momentuz, honakoa ondo funtzionatzen du, eta hurrengo faseari hasiera emango zaio, aplikazioari funtzionalitate gehiago gehituz.

04/24-04/29 testak

04/24 eta 04/29 daten artean garatutako funtzionalitateak probatu egingo dira. Hemen garatutako funtzionalitateak aurreko fasean garatutakoa asko zabaltzen du, beraz berriro ere honako funtzionalitatea sakonki probatu egingo da.

Probatu egingo diren funtzionalitateak honakoak izango dira:

1. Eskaera bat spam moduan markatu.
2. Eskaera baten laguntza onartu.
3. Eskaera baten laguntza ezeztatu.
4. Mezua bidali.
5. Chat berria sortu.

Erantzunen taula

Deskribapena	Espero den erantzuna	Lortutako erantzuna	**Oharrak**
Spam moduan markatu eskaera	Notifikazioa eta botoia argitu.	Spam moduan markatuta. Botoia argitu da eta notifikazioa erakutsi da.	**Eskaera datu basean dago, ez da borratu.**
Spam moduan markatu datu basetik borratu den eskaera.	Errorea, aurkitu ez dala esanez.	Errorea, eskaera existitzen ez dela esanez.	Denak lortzean bazegoen, ondoren eskaera borratu da eta gero markatu da spam moduan.
Laguntza onartu.	Mezu automatiko bat bidali eta eskaera desagertu.	Mezu automatiko bat bidali da. Eskaera ez da ikusgai beste erabiltzaileentzako.	Erabiltzaile batek baietz esatearekin nahikoa da.
Laguntza onartu (eskaera datu basetik borratu da)	Abisua, chat-a borratu dela esanez.	Abisua, chat-a borratu dela esanez.	
Laguntza onartu (erabiltzaileak kontua ezabatu du)	Abisua erabiltzailea ez dela aurkitu esanez.	Abisua erabiltzailea ez dela aurkitu esanez.	
Laguntza ezeztatu	Chat orritik irten eta chat hori borratu listatik.	Chat orritik irten eta chat hori borratu da listatik.	Beste erabiltzaileak mezu bat jasoko du chat hori borratu dela esanez.
Mezua bidali	Mezu berria bidaliko da.	Mezu berria bidaliko da.	
Mezua bidali (eskaera ezabatu da)	Eskaera aurkitu ez dela abisatu.	Eskaera aurkitu ez dela abisatuko da.	
Mezua bidali (erabiltzailea ezabatu da)	Erabiltzailea aurkitu ez dela abisatu.	Erabiltzailea aurkitu ez dela abisatu.	

B.6.taula: test erantzun taula

Probatutako funtzionalitate hauek espero zen erantzuna itzuli dutenez, fasea itxi egingo da eta hurrengoari hasiera emango zaio.

01/05-04/05 testak

Ez dira funtzionalitate berririk inplementatu fase honetan, baina aurkitu diren errore batzuk konpondu eta gero, hauek berriz probatu egin dira.

Errorea chat-ean mezua bidaltzean gertatzen zen, eta oso gutxitan. Honakoa berriz garatu egin da eta proba berriak egin dira.

Berriz probatuko diren funtzionalitateak honakoak izango dira:

1. Mezua bidali.
2. Laguntza onartu.
3. Laguntza ezabatu.

Erantzunen taula

Deskribapena	Espero den erantzuna	Lortutako erantzuna
Mezua bidali.	Mezua pantailan idatzi (behin)	Mezua pantailan idatzi da (behin)
Laguntza onartu.	Mezu automatiko bat bidaliko da eta eskaeraren datuak eguneratuko dira.	Mezu automatiko bat bidali da eta eskaeraren datuak eguneratu dira.
Laguntza ezeztatu.	Chat-a borratu egingo da.	Chat-a borratu egingo da.

B.7.taula: test erantzun taula

Proba hauen ondoren, funtzionalitateak zuen errorea konpondu egin dela eta guztiz funtzionala dela ondorioztatu daiteke. Beraz, honako fasea bukatutzat emango da.

C eranskina

Jarraipen eta kontrol txostena

Proiektuaren kudeaketa kapituluan (3. kapitulua) aipatu den bezala, proiektu honi 300 ordu dedikatu behar zitzaizkion. Ordu hauek banatu egin dira ataza desberdinetan eta bakoitzarentzako beharrezkoa izango den lan denbora estimazio bat egin da.

Ondorengo taulan, lan-pakete eta ataza bakoitzean estimatutako denbora, benetan dedikatu zaion denbora kopurua eta desbideraketak azaltzen dira.

LAN PAKETEA/ATAZA	ESTIMATUTAKO ORDUAK	ERABILITAKO ORDUAK	DESBIDERAKETA
Kudeaketa	**25 h**	**21h 30'**	**+3h 30'**
Plangintza	10 h	8h 30'	+1h 30'
Betekizunen azterketa	3 h	3h	-
Lan metodologia	2 h	1h 30'	+30'
Teknologiak aukeratu	5 h	4h	+1h
Jarraipena eta kontrola	10 h	10h 30'	-30'
Bilerak	5 h	2h 30'	+2h 30'
Produktua	**175 h**	**176h 55'**	**-1h 55'**
Teknologia ikasketa	10 h	15 h	-5h
Garapena	165 h	161h 55'	+3h 5'
Diseinua	12 h	13h	-1h
Datu basea	2 h	1 h	+1h
Maketa	10 h	12 h	-2h
Inplementazioa	133 h	131 h 55'	+1h 5'
Testak	20 h	17 h	+3h
Dokumentazioa landu	**100 h**	**94 h 40'**	**+5h 20'**
Memoria idatzi	70 h	64 h 40'	+5h 20'
Gardenkiak landu	30 h	30 h	-
Guztira:	300 h	293h 5'	+6h 55'

C.1.taula: lan-pakete eta atazen denbora konparaketak

3.kapituloko jarraipenaren emaitzak atalean azaldu den bezala, proiektuan zehar gertatutako desbideraketa ez da hain handia izan. Hauek jasangarriak diren desbideraketak dira eta ondoren garapenean egon den denbora soberakina dela eta, ez du eraginik izan proiektuan.

Proiektuaren barnean hainbat data garrantzitsu ere definitu egin dira. Alde batetik proiektuarekin orokorrean definitutako epemugak daude eta bestetik garapen barruan definitutako *sprint*-ei definitutako datak. Jarraian agertzen diren taulak honako datuak azaltzen ditu izandako desbiderapenarekin.

Deskribapena	Estimatutako data	Benetako data	Desbiderapena
Proiektuaren hasiera	03/21	03/21	-
Aplikazioaren lehen bertsioa	05/12	05/08	+4 egun
Memoriaren lehen bertsioa	05/12	05/18	-6 Egun
Aplikazio definitiboa	05/26	05/13	+13 egun
Memoriaren azkeneko bertsioa	06/09	06/14	-5 egun
Proiektuaren itxiera	06/09	06/14	-5 egun
Proiektuaren defentsa	07/10-13	07/10-13	-

C.2.taula: mugarrien konparaketa

Aipatu beharra dago, garapenean izandako desbideraketak ez duela produktuaren lehenengo bertsioan eta azkenekoan atzerapenik ekarri. Aldiz, hauek behar baino lehenago ateratzea lortu egin da. Beraz, nahiz eta garapenerako beharrezkoa zen denbora egokia ez izan, epemugak ondo kalkulatu egin dira.

Nabarmena den desbideraketa memoriarekin aurkitzen da. Lehenengo eta azkeneko bertsioaren artean, tutoreak lana zuzendu behar du eta egiten dituen iradokizunak eta aldaketak aplikatzeak bere denbora hartzen du. Ala ere, dokumentua irakurtzeko beharrezkoa izango duen denbora estimatzea ez da posible, beraz hauetan desbiderapen bat egotea ere jasangarria da.

Garapena 8 *sprint* desberdinetan banatu egin dela aipatu egin da. Jarraian bakoitzaren desbideraketa konparaketa.

Sprint identifikazioa	Estimatutako data tartea	Egindako data tartea	Desbideraketa
SB-1	03/21-03/24	03/21-03/24	-
SB-2	03/27-03/31	03/27-03/31	-
SB-3	04/03-04/07	04/03-04/06	+1 egun
SB-4	04/10-04/14	04/10-04/13	+1 egun
SB-5	04/17-04/21	04/14-04/23	-5 egun
SB-6	04/24-04/28	04/24-04/29	-1 egun
SB-7	05/01-05/05	05/01-03/05	+2 egun
SB-8	05/08-05/12	03/08-05/13	-1 egun

C.3.taula: sprint-en daten konparaketa

Hemen *SB-5 sprint*-a izan da desbideraketa handiena jasan duena. Atal honetan aplikazioak dituen funtzionalitate zailenetako bat, chat-aren funtzionalitatea zehatzagoak izateko, inplementatu da. Gainera, aurkitutako errore batzuk direla eta, hemen behar baino denbora gehiago inbertitu behar izan zan da, garapen denboran desbideraketa handia sortuz.

D eranskina

Erabiltzaileen ebaluazio txostena

Erabiltzaileei bi motatako ebaluaketa egin zaie. Alde batetik aplikazioa probatzea eskatu zaie, eta beraien iritzia eman zezaketen. Proba hauetan erabiltzaileak aplikazioa nola erabiltzen zuten aztertu egin da (zalantzak izan dituzten, gauza bat egitean aurkitu duten, …) eta ondoren aplikazioan zer bota duten faltan galdetu zaie.

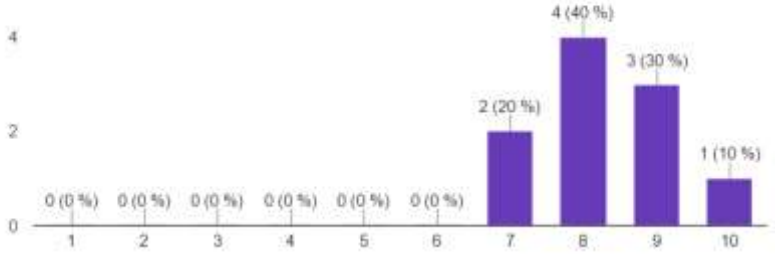
Aplikazioa 10 pertsonek probatu egin dute. Hauek sexu, adin eta informatika ezagutza maila desberdinekoak izan dira.

Aplikazioa probatzen

Google formulario bat sortu egin da aplikazioarekin bukatzean beraien esperientzia ebaluatu zezaten. Jarraian galderak eta jasotako erantzunak:

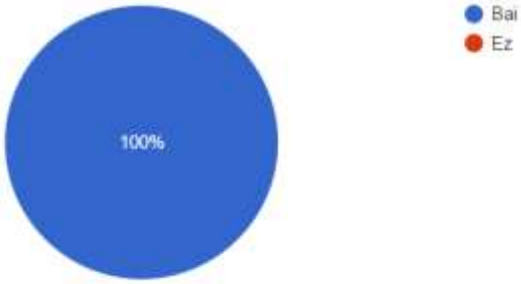
Puntuatu beharko bazenu, ze nota emango zenioke?

10 respuestas



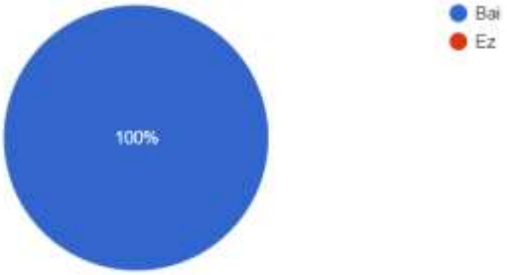
Bere funtzionamendua erreza iruditu zaizu?

10 respuestas



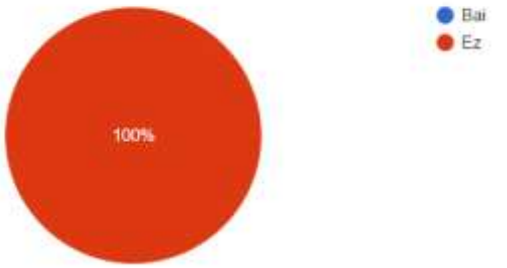
Egin nahi ziren gauzak modu azkarrean egin dira?

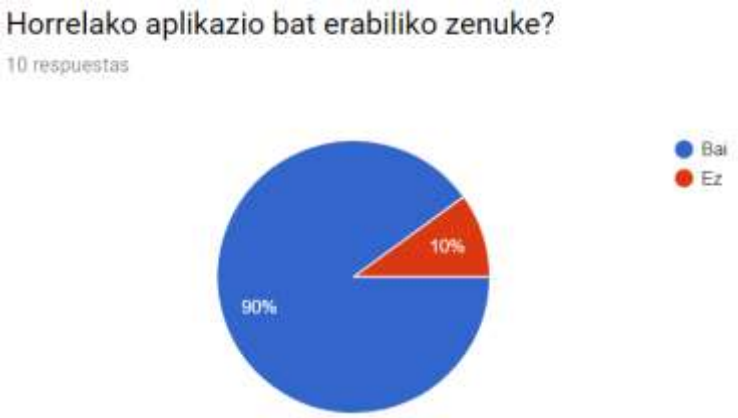
10 respuestas



Momenturen batean zaila ikusi duzu zerbait aurkitzea?

10 respuestas





D.1.irudia: ebaluazio formularioaren erantzunak

Probatzeko utzi zaienean aplikazioa, berarekin lanean segituan hasi direla aipatu behar izan da. Behin erregistroa bete dutenean, libre utzi zaie, eta funtzionalitate guztiak betetzen zuten aztertu egin da. Ez da beharrezkoa izan inongo azalpenik. Irekitzen zutenean denek esaten zuten Instagram edo Twitter ematen zuela, beraz interfazea nahiko ezaguna zutela.

Erantzunak ikusita, ikusten da modu pertsonalean aipatutakoa bete egiten dela. Interesa ere handia dela ikusten da, galdetutako pertsonen %90-ak erabiliko lukeela erantzun baitu (beste %10-a ikaslea ez delako erantzun du ezetz).

Hobekuntza posibleak

Formulario berdina erabilita, zer nahi zuten jar zezaketen. Honakoak izan dira jarritako erantzunak, batzuk errepikatuta zeuden eta beste batzuk atal hau hutsa utzi egin dute.

- Chat-ean gauza gehiago bidaltzea (irudiak, fitxategiak eta audioa).
- Beste motatako eskaerak egitea. Adibidez kotxea partekatzeko.
- Perfiletan irudiak gehitu.
- Notifikazioak pantaila blokeatuta dagoenean (*push*).

Hobekuntza hauen inguruan gauza gehiago jakiteko, 9.kapitulua.

E eranskina

Domeinuaren diseinua

Atal honetan SailsJS ingurunean datu base bat nola sortu den adierazten da. Hau egiteko proiektu honek erabiltzen duen datu basea nola sortu egin den azalduko da.

Datu basearen konfigurazioa izango da web zerbitzuarekin egingo den lehenengo gauza. Konfigurazio bat egin behar denez, konfigurazioa egiten den karpetara jo behar da. Aplikazioaren konfigurazioarekin zerikusia duen edozer gauza `Config` karpeta barruan dauden fitxategietan egiten da. Karpeta honen barruan datu basearekin konexioa, socket-en konfigurazioa eta helbideen konfigurazioa aurkitu daitezke besteak beste.

Datu base baten konfigurazioa egiteko lehenengo zein datu base instalatu nahi den erabaki behar da. Aipatu egin da aplikazio honek MySQL datu base bat erabiliko duela, beraz honetan oinarrituta datu base baten konfigurazioa nola egiten den azalduko da jarraian.

1. `npm install sails-mysql` komandoa exekutatu komando lerroan, proiektuan MySQL instalatzeko. Honek beharrezkoak diren loturak eta paketeak deskargatu eta sortuko ditu.
2. `connections.js` datu basea sortu. Horretarako fitxategi horretan honakoa idatzi:

```
mysql: {
  adapter: 'sails-mysql',
  host: 'localhost',
  user: 'root', //optional
  password: 'root', //optional
  database: 'lexa' //optional
},
```

E.1.irudia: connections.js fitxtegian MySQL konfigurazioa

Hainbat gune editatu beharko dira gure datu basearekin bat egiteko. Jarraian atributu bakoitza zertarako den.

- Host: datu basea non aurkitu den adierazten duen atributua. Garapen prozesuan ordenagailuan (lokalki) aurkitu ohi da, beraz `localhost` jarriko da. Datu basea beste leku batean kokatuta badago, kanpo zerbitzari batean adibidez, horren IP helbidea jarriko da.
- User: datu basearen erabiltzailea zein den zehaztu behar da. Honakoa hautazko eremua izango da.
- Password: datu basearen pasahitza. Honakoa ere hautazko eremua izango da.
- Database: datu basearen izena zein izango den zehazten duen eremua. Honakoa hautazkoa da, baina izena jartzea gomendatzen da ondoren datu basearen identifikazioa egin ahal izateko.

Konexioari izen bat gehituko beharko zaio **“:{…}”** baino lehenago, kasu honetan konexioaren izena `mysql` izango da baina nahi den izena eman daiteke. Datu base bakarrarekin lan egiten denean konexioari izen bat ezartzeak ez du garrantzi handirik, baina datu base batekin baino gehiagorekin lan egiten denean, konexio bakoitza desberdintzea garrantzitsua izango da.

3. `models.js` fitxategian sortuko diren ereduak zein datu basetan gorde nahi diren konfiguratzeko fitxategia.

```
module.exports.models = {
  connection: 'mysql',
  migrate: 'alter'
};
```

E.2.irudia: models.js fitxategiaren konfigurazioa

Hemen, bi izango dira zehaztu beharreko atalak:

- Connection: konexioetan sortu den datu basearen konexioaren izena. Konexio hau erabilita datuak gordeko direla adieraziko da. Lan honetan konexioari `mysql` izena eman zaio, beraz balio hori adieraziko da hemen.
- Migrate: zerbitzaria martxan jartzen den bakoitzean, ereduetan aldaketak egin diren aztertuko da. Hemen aldaketa hauek nola exekutatuko diren adierazten da. Jar daitezkeen balioak honakoak izan daitezke:
    - Safe: datu basea ez migratu. Egin beharreko aldaketak garatzaileak egingo ditu eskuz.
    - Alter: automatikoki eguneratu datu basea, sortu edo aldatu beharreko zutabeak edo eremuak eguneratuz. Datu basean zeuden datuak mantendu egingo dira.
    - Drop: datu base osoa borratu egingo du eta berriz sortuko da. Hau egiten bada, datu basean zeuden datuak ezabatu egingo dira.

4. Komando lerroan `sails lift` exekutatu eta egiaztatu ez daudela errorerik. Errore mezuren bat badago, errepasatu sartutako datuak.

F eranskina

Mezu maketatuak

SailsJS lan inguruneak mezu maketatuak bidaltzea ahalbidetzen du. Eranskin honetan hau nola egiten den azalduko da.

Webgune batzuetan erregistratzean edo aldaketa garrantzitsuak egiten direnean, korreo bat iritsi ohi da, segurtasun modu bat kontsideratu daiteke. Niri korreoa iristen bazait eta ez banaiz bertan erregistratu, norbait nire korreoa erabiltzen dagoela edo nahigabe idatzi duela jakingo dut. Horregatik, honakoa egiten duen funtzionalitatea gehitzea erabaki da.

Mezuak modu automatikoan bidaltzea zerbitzariaren lana da. Bi kasu zehatzetan soilik egitea erabaki da: erregistroa egiten denean (sartutako datuak erabiltzaileari jakinarazteko) eta kontua borratzean (kontu hori ezabatu dela adierazteko).

Honakoa egiteko, lehenengo pausoa aplikazioarentzako posta kontu bat sortzea izango da. MailGun, Gmail edota beste posta zerbitzu bat aukeratu daiteke. Kasu honetan Gmail

zerbitzua aukeratu da, Google-ek ematen dituen beste zerbitzu batzuk ere erabili daitezkeelako (formularioak, Drive, Docs, …).

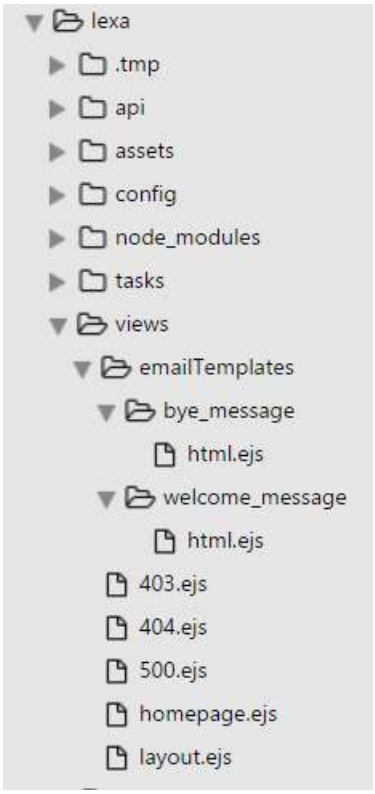
Behin kontua sortuta, zerbitzaria posta baten konfigurazioa egiteko prestatu beharko da. SailsJS-rako GitHub-en aurkitutako repositorioa[30] erabili da. NodeJS-rako den NodeMailer moduluan oinarritzen den modulu bat da baina SailsJS lan ingurunera moldatuta. Instalazioa egiteko `npm install sails-hook-email` exekutatu beharko da komando lerroan (proiektua dagoen direktorio berdinean gaudela ziurtatu).

Postaren konfigurazioa egiteko beharrezkoa den fitxategia ez da defektuz existitzen eta moduluaren instalazioak ez du fitxategi hau sortzen. Horregatik, `config` karpeta barruan, `email.js` izena duen fitxategi bat sortu beharko da. Fitxategi honetan sortu berri den posta horren datuak definitu beharko dira, hau da, erabiltzaile izena, pasahitza, zein zerbitzu erabiltzen den, etab.

```
module.exports.email = {
    service: "Gmail",
    auth:{
        user: "lexaapplication@gmail.com",
        pass: "********"
    },
    testMode: false,
};
```

F.1.irudia: email.js fitxategiaren konfigurazioa

Bidaltzen den posta itxura polita izatea nahi da, hau da, testu soila ez izatea. Horrela ere benetan aplikaziotik bidalitako zerbait dela ikusiko da. Beraz, postaren maketazioa egin beharko da. Fitxategi hauek EJS (JavaScript-en txantiloiak sortzeko luzapena) fitxategiak izango dira eta bezeroei mezua bidaltzen zaienean bidaliko den fitxategia izango da. Zerbitzariaren egitura jarraituz, fitxategi hauek bistak izango dira, beraz `views` karpeta barruan egon beharko dira. Txantiloiekin lan egiten denez, komeni da atalka banatzea bista bakoitza. Orain postarako txantiloiak sortzen direnez, `emailTemplate` deitzen den karpeta bat sortzea gomendatzen da, postarekin zerikusia duten fitxategi guztiak leku berean egon daitezen. Ala ere, bi kasutan posta bat bidaliko dela esan da, beraz bi txantiloi desberdin sortu beharko dira. Hauentzako ere karpeta desberdinak sortuko dira, ondoren mezua bidaltzean zein txantiloi nahi den adieraztea errazagoa izango baita. Ondoren, sortu berri diren karpeta hauetan txantiloiak (EJS fitxategiak) sortu behar dira. Hauei `html.ejs` izena eman beharko zaie ondoren bista modu automatikoan har dezan.



F.2.irudia: views karpetaren egitura

Konfigurazioa eta maketa prest dugula, korreoa benetan bidali beharko da. Hau kontrolagailuak egin beharko du dagokion funtzioaren barruan. Kasu honetan erabiltzaile erregistratzean (`UserController`-eko `index` funtzioan) eta hau ezabatzean (`UserController`-eko `delete` funtzioan) exekutatuko da. Mezua bidaltzeko `sails.hooks.email.send()` funtzioa erabili beharko da dagozkion atributuak emanez:

1. *Template* edo postaren txantiloia. Aurrerago aipatutako EJS fitxategiak zein karpetaren barruan dauden azaldu beharko da. Modu automatikoan `html.ejs` fitxategia hartzen du (`bye_template` karpeta adibidez).
2. Datuak. Txantiloiak erabiltzen direnez, postak bata bestetik desberdinak egin daitezke. Adibidez batek “Kaixo Xabin” eta beste batek “Kaixo Ane” bidali dezake. Hemen adierazi beharko dira bata bestetik desberdinak izango diren datuak.
3. Korreoaren konfigurazioa. Hemen zein posta elektronikora eta zein asuntorekin bidaliko den adierazi behar da.
4. Mezua bidali eta gero exekutatuko den kodea, behin mezua bidalita zerbait egin nahi bada.

```
sails.hooks.email.send(
    "welcome_message",
    {
        username: user.name,
        email: user.email,
        password: user.password
    },
    {
        to: user.email,
        subject: "Ongi etorri!",
    }, function(err){
        if(err)
            return res.serverError(err);
        else
            return res.ok();
    }
);
```

F.3.irudia: ongi etorria bidaltzeko mezuaren funtzioa User.js barruan

Erabiltzaileek jasoko duten korreoa sortutako korreoa eta asuntoarekin iritsiko da.

F.4.irudia: posta maketatua

G eranskina

Mantenua

Zerbitzari baten mantenua egiten denean aplikazioa mantentzen duen administratzailea edo gestorea den pertsona bat ezarri ohi da. Pertsona honek datuak garbitzeaz arduratzen da eta datu basea oso beteta dagoenean, hau ustutzeaz.

Hemen administratzailearen rol hau ez da existitzen, dena automatizatuta egotea erabaki da. MySQL Workbench programari esker, datu base baten gainean gertaerak definitu daitezke. Hauek *trigger*-en antzekoa da, baina data bat edota denbora tarte bat definitzean jaurti egiten diren funtzioak (edo funtzio multzoak) dira. Gertaera hauek soilik zerbitzaria martxan dagoenean gertatuko dira, hau da, gertaera bat goizero 11:00-tan exekutatu nahi bada eta zerbitzaria itzalita badago, ez da gertaera hori aplikatuko. Aldiz, piztuta badago, adierazitako funtzioak exekutatuko dira.

Datu basea garbitzeko erabaki den maiztasuna egun batekoa izango da, hau da, 24 orduro gertaera hau modu automatikoan exekutatuko da. Borratu beharreko datuak kasu honetan

erabiltzaileek adierazitako epemuga betetzen duten eskaerak (eta berarekin erlazionatutako chat guztiak) ezabatzea izango da.

```
drop event if exists delete_petitions;

delimiter |

CREATE EVENT delete_petitions
ON SCHEDULE
       EVERY 1 DAY
    STARTS '2017-04-13 00:00:00' ON COMPLETION PRESERVE ENABLE
DO
       BEGIN
              # Ezabatu chat taulatik
              DELETE FROM lexa.chat
              WHERE chat.petition_id = (
                 SELECT petition.id
                 FROM lexa.petition
                 WHERE
                 STR_TO_DATE(petition.deadline, '%Y-%m-%d') < CURDATE());

              # Ezabatu petition taulatik
              DELETE FROM lexa.petition
              WHERE
              STR_TO_DATE(petition.deadline, '%Y-%m-%d') < CURDATE();

       END |
```

G.1.kodea: eskaerak automatikoki borratzeko gertaera

Baina gertaerak exekutatzeko zerbitzaria martxan egoteaz gain, MySQL zerbitzariak aldagai globaletan **event_schedule** deritzon atributua aktibatuta egon behar du. Horretarako, MySQL Workbench erabiltzen bada, G.2.kodea (a) exekutatzearekin nahikoa izango da. Honek ematen digun erantzuna ON edo OFF izango da. OFF jartzen badu, G.2.kodea (b) exekutatu beharko da aktibatzeko.

```
SHOW VARIABLES
WHERE
VARIABLE_NAME = 'event_scheduler'
```

(a)

```
SET GLOBAL event_scheduler =
ON;
```

(b)

G.2.kodea: gertaeren informazioa lortzeko kodeak