Universitat Oberta de Catalunya

uoc.edu

Sistema de recompenses i lleialtat

Sergi Montañés Diaz-Tejeiro
Grau d'Enginyeria Informàtica
Treball de Final de Grau - Java EE

Albert Grau Perisé
Santi Caballe Llobet

12/06/2020

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement 3.0 Espanya de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Copyright © ANY EL-TEU-NOM.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free

Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright

© (l'autor/a)
Reservats tots els drets. Està prohibit la reproducció total o parcial d'aquesta obra per qualsevol mitjà o procediment, compresos la impressió, la reprografia, el microfilm, el tractament informàtic o qualsevol altre sistema, així com la distribució d'exemplars mitjançant lloguer i préstec, sense l'autorització escrita de l'autor o dels límits que autoritzi la Llei de Propietat Intel•lectual.

FITXA DEL TREBALL FINAL

Títol del treball:	*Sistema de recompenses i lleialtat*
Nom de l'autor:	*Sergi Montañés Diaz-Tejeiro*
Nom del consultor/a:	*Albert Grau Perisé*
Nom del PRA:	*Santi Caballe Llobet*
Data de lliurament (mm/aaaa):	*06/2020*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Treball de Final de Grau - Java EE*
Idioma del treball:	*Català (part tècnica amb anglès)*
Paraules clau	*Plataforma Recompenses Lleialtat*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Amb aquesta projecte, s'ha volgut oferir un servei de lleialtat i recompenses, que permeti als usuaris/compradors, acumular punts en base a diverses accions, per després canviar-les per recompenses que les diferents organitzacions oferiran. Enfocat a restauració, però no limitat, podent comprendre qualsevol sector que estigues interessat en un sistema com aquest

Implementant una arquitectura orientada a serveis, ens permet el poder separar totalment el client de la part servidora, i guanyar en escalabilitat i distribució. Per tant, el nucli de plataforma es basa en una interfície de programació d'aplicacions (en anglès Application Programming Interface, API), oferint els serveis, i en un client web, que serà el que les diferents organitzacions faran servir, i que es comunicarà directament amb el nostre sistema.

S'ha pogut presentar un producte mínim viable que es operatiu, i interessant, oferint el servei que es volia, però havent retallat en alguns aspectes de gestió , i deixant fora un tercer component, el qual era el client per a mòbils.

Això s'ha degut a que en la planificació no es va incloure un marge de temps extra, per el que, al haver certs aspectes que es van allargar o que no estaven previstes, aquests han impactat directament amb els temps d'entrega.

Abstract (in English, 250 words or less):

With this project, we wanted to offer a service of loyalty and rewards, which allows users/buyers to accumulate points based on some actions, and then exchange them for rewards that the different organizations will offer. The main target of this product is hospitality businesses, but not restricted.

Implementing a service-oriented architecture allows us to totally separate the frontend from the backend and gain in scalability and distribution. Therefore, the platform core is based on an API, providing all services, and a web client, which will be the one used by the different organizations, and which will communicate directly with our system.

It has been possible to present an MVP that is functional and interesting, achieving the service that we wanted, but having to cut out on some aspects of management, and leaving out a third component, which was the mobile client.

This has been because the planning did not include a buffer time, resulting in a delay on some deadlines, as there were certain aspects that were extended or not foreseen. This has had a direct impact on the final deadline.

Índex

1. Introducció 1
1.1 Context i justificació del Treball 1
1.2 Objectius del Treball 2
1.3 Enfocament i mètode seguit 3
1.4 Planificació del Treball 3
1.5 Breu sumari de productes obtinguts 5
1.6 Breu descripció dels altres capítols de la memòria **¡Error! Marcador no definido.**
2. Anàlisi Funcional 6
2.1 Requeriments 6
2.2 Actors 8
2.3 Casos d'us 9
3. Disseny 17
3.1 Diagrama de seqüencia 17
3.2 Disseny relacional de la base de dades 18
3.3 Diagrama de classes principals 21
3.4 Disseny de l'arquitectura del sistema 22
3.5 Disseny de la interfície d'usuari 23
4. Implementació 29
4.1 Eines i frameworks 29
4.2 Seguretat 31
4. Conclusions 33
4. Glossari 34
5. Bibliografia 36
Articles electrònics: 36
Webs de referencia: 36
6. Annexos 37
Manual execució Rewards System 37

1. Introducció

1.1 Context i justificació del Treball

En un moment on els mercats son tant competitius, els consumidors aprecien enormement la millora de la interacció amb una marca, impulsada per les recompenses i el reconeixement per completar diverses accions (per exemple, visitar una botiga en particular, fer una compra, proporcionar informació sobre el producte). Això proporciona als establiments un valor afegit i diferencial, i permet consolidar una posició de marca en el mercat.

Per això, la finalitat d'aquest projecte es el d'oferir una plataforma de lleialtat i recompenses per a negocis, especialment de model franquícia, però també per a indústries rellevants com el comerç minorista, el turisme regional, els esdeveniments i el màrqueting de recintes.

Aquesta plataforma permetrà als usuaris/compradors, acumular punts en base a diverses accions, per després canviar-les per recompenses que les diferents organitzacions oferiran.

En lloc de simplement resoldre el problema amb una aplicació ad hoc per a un únic comerç, la idea del projecte es desenvolupar una API que oferirà aquest servei, públic per als clients i accessible a traves del *cloud*, i un seguit d'aplicacions client que pivotaran al seu voltant, sense interferir les unes amb les altres. Això permetrà comprendre diferents sectors i geografies, que a simple vista poden semblar molt diferents, simplement personalitzant o creant noves aplicacions clients, sense tenir que tocar el seu nucli, la *API*, que serà el centre neuràlgic de tot el sistema.

1.2 Objectius del Treball

Com aquesta aplicació te com a objectiu el oferir un servei a mes d'una organització, es de vital importància per al sistema el ser escalable, permetent la inclusió de noves organitzacions al llarg del temps, sense que això impacti de manera negativa en el rendiment o que l'arquitectura no sigui un impediment.

Al tenir com a objectiu múltiples negocis i usuaris, hi ha varis components que entraran en joc:

- La part mes gran serà la *API*, utilitzant el *cloud* per allotjar-la i ser accessible des dels diferents clients. Serà l'encarregada de gestionar els punts, usuaris, recompenses, etc

- Client de l'API per part de les diferents organitzacions. Ens permetrà llegir les targetes dels compradors, i gestionar les diferents recompenses i usuaris.

Com a tercer component, el qual no s'ha desenvolupat en aquest projecte, tenim:

- Client de l'API per part dels compradors. Tradicionalment, han sigut sempre targetes físiques(les quals hem mantingut per simplificar el projecte i complir amb les dates d'entrega), però sent que cada any, la quota de mercat dels mòbils creix, es pretén realitzar una aplicació mòbil per als usuaris, on es podran consultar els punts actuals i ofertes disponibles, i el mes important, contindrà la targeta digital.

Com que l'abast del projecte era massa gran, el tercer punt s'ha mantingut fora, i passat a utilitzar únicament targetes físiques. Al ser que ambos sistemes compatibles, el tradicional amb la targeta física i l'*app* amb la targeta digital, podran conviure perfectament en cas de que en un futur volguéssim implementar aquest punt i desenvolupar la *app*.

Això ens demostra l'escalabilitat a nivell de negoci d'aquest projecte, on es poden incloure nous clients que giraran al voltant del servei principal, la API, sense interferir les unes amb les altres.

1.3 Enfocament i mètode seguit

Per aquest projecte, hem desenvolupat de base la *API*, incloent les funcionalitats bàsiques, com son les gestions d'usuaris, punts i recompenses, i sobretot, seguretat, que es el que ens permetrà tenir diverses organitzacions en el sistema, treballant de manera concurrent.
El motiu d'un desenvolupament nou, es perquè aquests tipus de sistemes son, per norma general, tancats. Com a exemple, tenim els supermercats, els quals fan servir un sistema similar, però es únic i personalitzat per aquella cada supermercat. Una plataforma d'aquest tipus allotjada al *cloud* i auto definida com multi-organització, es relativament poc comú e innovador.

El client s'ha desenvolupat pensant en empreses de restauració, ja que, tot i que la nostra plataforma aspira a ser extrapolable a altres sectors, si agafem com a referencia altres països on aquesta es una practica mes habitual, aquest es el que tindria mes tirada per a un sistema com el nostre.

1.4 Planificació del Treball

El desenvolupament de la part tècnica d'aquest projecte s'ha portat a cap en un període de 3 mesos aproximadament(Març a Juny), i s'ha seguit escrupolosament per així poder tenir un producte mínim viable per a principis de Juny.

A grans trets, les principals tasques que s'han inclòs en aquesta planificació, juntament amb les dates proposades, son:

- <u>Esquema DB</u>: Creació de tota l'estructura relacional de Base de dades, incloent *scripts* amb dades de proba, per poder portar a terme les probes funcionals. Aquesta part inclourà doncs la instal·lació del gestor PostgreSQL, i la creació de diversos scripts SQL, amb la creació d'usuaris, tablespace, base de dades, taules, privilegis per a l'usuari, i *inserts* per les dades de proba.
    - 2 March 2020 – 6 March 2020
- <u>Creació de les bases de l'aplicació SpringBoot</u>: Creació de les bases de la *API*. Incloure les dependències bàsiques, com ara *Spring*, i adoptar un sistema per automatitzar la compilació i empaquetar el codi font i dependències en un executable, fent servir Gradle.
    - 9 Març 2020 – 16 Març 2020

- Creacio del model amb ORM: Creació del model de la nostra API basant-nos en l'estructura relacional creada anteriorment, i implementació del *framework Hibernate* pel mapeig objecte - relacional(*ORM*), sobre la nostra base de dades, i així facilitar les tasques de persistència, a demes de dotar a la nostra aplicació de portabilitat en cas de voler canviar de gestor de base de dades.

    - 17 Març 2020 – 25 Març 2020

- Creació del punts d'accés: Creació dels Controladors i serveis, definint així els punts d'entrada i la lògica de negoci, desenvolupats pensant en els requeriments del sistema de l'apartat 2.1.

    - 16 Març 2020 – 24 Març 2020

- Implementació del JWT: Implementació del *login* i la seguretat, fent us de la dependència Spring Security, per a poder filtrar totes les peticions provenint del client, i validar-les, en busca d'un *token JWT* vàlid, o en el cas del *login*, crear-lo.

    - 26 Març 2020 – 10 Abril 2020

- Creació de les bases del Client: Maquetació i creació dels estils *css*. Construcció inicial del client fent servir *javascript* i la llibreria *JQuery*.

    - 7 Març 2020 – 22 Març 2020

- Lector QR: Investigació de llibreries *javascript* que permetin la creació i lectura de codis QR, i que encaixin en el nostre cas. Implementació d'aquestes en el nostre projecte.

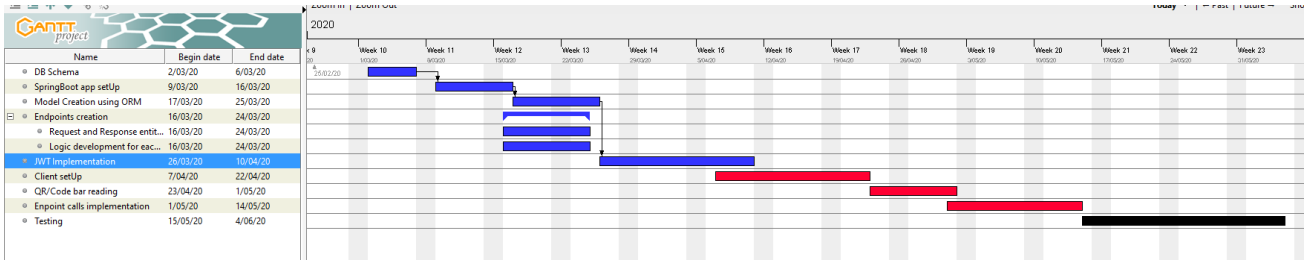
    - 23 Abril 2020 – 1 Maig 2020

- Implementació de la crida als punts d'accés: Implementació de la comunicació asíncrona *Ajax* entre el client i el servidor, afegint la resta de lògica del client amb *javascript*.

    - 1 Maig 2020 – 14 Maig 2020

- Integració i probes: Probes funcionals i d'integració entre el client i el servidor, executant els diferent casos d'us, i les correccions necessàries .

    - 15 Maig 2020 – 4 Juny 2020



Com a grans fites a assolir, tenim la part de la *API*, sent el 10 d'Abril com a límit d'entrega, i la part del client, a entregar el 15 de Maig.
Una vegada finalitzades les dos *apps*, s'entra a la fase de *testing*, amb 15 dies per executar les probes funcionals, basades en els diferents casos d'us que tenim.

1.5 Breu sumari de productes obtinguts

La plataforma de recompenses desenvolupada esta dividida en dos part, client i servidor.

El servidor, el qual es una API, inclou totes les funcionalitats de pagaments, usuaris i organitzacions necessàries per al producte mínim viable. També inclou la seguretat, creant tokens JWT per cada organització registrada que accedeix al sistema, i afegint validacions per vetar les peticions no autoritzades. La API ofereix serveis a tots els programes clients, tant al que hem desenvolupat al llarg d'aquest projecte, com qualsevol altre que pogués ser desenvolupat en un futur.

Les funcions que s'han deixat a fora d'aquest producte mínim viable son:

- Alta usuaris(Es comptarà amb usuaris precarregats en el sistema)
- Eliminar ofertes(En aquest cas, s'ha volgut deixar fora en la versió presentada, degut a que l'històric de l'usuari va lligat a ofertes, i eliminar una afectaria a la integritat referencial)
- Editar ofertes(Mateixa raó. Editar una oferta i els seus punts, podria alterar el històric, que es basa en els punts de les ofertes.)

Com es pot apreciar, s'ha buscat el tenir una funcionalitat de cada, ja sigui en ofertes o usuaris, es a dir, CRUD(create, read, update and delete, o en català, crear, llegir, actualitzar i borrar)

El client, el qual es web, compte amb totes les funcions integrades a la API, a part de totes les funcions de lectura i generació de codis QR.

No s'ha pogut entregar una tercera part, que es la conversió del client a *mobile*.

2. Anàlisi Funcional

2.1 Requeriments

Les funcions a implementar es poden separar en quatre grans blocs, basats en el *login*, la gestió d'usuaris, gestió d'ofertes i pagament:

Login

Ens permetrà accedir a l'aplicació, amb un usuari únic associat a l'organització. Els camps per accedir, seran correu electrònic i contrasenya.

Gestió d'usuaris

- Registre d'usuaris a l'aplicació. Les empreses podran donar d'alta usuaris a traves d'un formulari que es trobarà en el client. El formulari constarà d'uns pocs camps, que seran Nom, cognom, email, i numero de telèfon(opcional). Si els establiments ho trobessin oportú, es podrien afegir mes camps en un futur.

- Edició d'usuaris. Les empreses podran donar editar usuaris a traves d'un formular, similar al de registre. En aquest formulari, els camps que es podran editar seran els de informació de l'usuari, es a dir, nom, cognom i telèfon.

- Eliminar usuaris. Es podran donar de baixa usuaris del sistema en aquella organització.

- Detalls d'usuari. Per cada usuari registrat, es podrà veure la seva informació en detall, com ara els punts que ha acumulat, el historial de punts, i el seu codi QR associat, que l'identificarà. Una vegada registrat, el establiment podrà accedir a aquest codi, per imprimir-lo sobre targetes.

Gestió d'ofertes

- Alta d'ofertes. A traves del client, es podran donar d'alta ofertes. Aquesta gestió es podrà fer des del programa client, i els camps dels que constaran aquestes ofertes seran nom i descripció, cost en punts, i data d'expiració. En el futur, es podria augmentar, per personalitzar mes les ofertes, afegint *banners* específics, o imatges de fons.

- Edició d'ofertes. Les empreses podran editar ofertes a traves d'un formular, similar al d'alta.

- Detalls de l'oferta. Per cada oferta donada d'alta, es podrà veure la seva informació en detall.

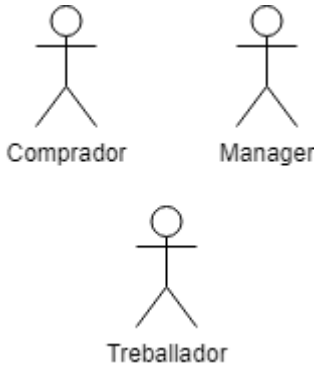
Pagament

- Lectura de codis QR. Al fer el pagament, l'establiment farà una lectura del codi QR del comprador, per identificar-lo, i passar a fer la acumulació o subtracció de punts. La targeta de l'usuari es llegida per el programa client, idealment instal·lat en un *smartphone*, i que farà us de la càmera per fer la lectura.

- Acumulació i subtracció de punts al comprador. El flux de treball principal serà l'acumulació de punts, on l'usuari presenta la seva targeta, que es llegida per el programa client juntament amb l'import que s'ha pagat, i l'oferta que s'ha usat, en cas de que hi hagi alguna. Aquesta informació es transferida a l'API, amb el codi de la targeta, que servirà per identificar al comprador, el codi del client, que serveix per identificar a l'establiment, i l'import, amb el que es calcularà el total de punts a acumular, i a sostreure, en que en el cas de que l'usuari faci us d'una oferta.

2.2 Actors

En el nostre sistema, es poden identificar diversos actors, com son:



- El comprador, que serà el client de l'establiment
- Manager de l'establiment, encarregat de publicar ofertes
- Treballador de l'establiment, encarregat de cobrar al comprador

A nivell de sistema, treballador i manager estan agrupats dintre de la mateix organització, per lo que serà una única compta d'usuari.

2.3 Casos d'us

Login

El login ens permetrà accedir a l'aplicació client fent us d'un compte d'organització o empresa, amb un usuari únic associat a l'organització. Els camps per accedir, seran correu electrònic i contrasenya.

Nom	A
Id	CU_00
Actor	Treballador establiment/Manager
Precondició	-
Seqüencia Normal	1. L'usuari es dirigeix a la pantalla de login 2. L'usuari introdueix correu i contrasenya, i prem el boto "Accedir" 3. El sistema redirecciona a l'usuari a la pagina index, on es mostren les diferents seccions a accedir.
Postcondició	L'usuari ha accedit a l'aplicació client, i totes les subseccions son visibles.
Excepció	**EX1. L'usuari falla al identificar-se** 1. El sistema notifica amb un missatge que les credencials introduïdes no son les correctes 2. El sistema manté a l'usuari en la mateixa pagina de *login*

Registre d’usuaris a l’aplicació per part de les empreses

Les empreses podran donar d’alta usuaris a traves d’un formulari que es trobarà en el client. El formulari constarà d’uns pocs camps, que seran Nom, cognom, email, i numero de telèfon(opcional). Si els establiments ho trobessin oportú, es podrien afegir mes camps en un futur.

Nom	Registre d’usuaris a l’aplicació per part de les empreses
Id	CU_01
Actor	Treballador establiment
Precondició	Treballador ha iniciat la sessió prèviament com a empresa.
Seqüencia Normal	4. El treballador es dirigeix a la pantalla de gestió d’usuaris 5. El treballador prem el boto de “Donar d’alta usuari” de l’aplicació client. 6. Es mostra per pantalla el formulari de alta. 7. El treballador emplena els camps “email”, “Nom” i “Cognom”, i prem el boto “Donar d’alta” 8. El sistema guarda les dades de l’usuari. 9. El sistema redirecciona al treballador a una pantalla de confirmació d’alta.
Postcondició	L’usuari s’ha guardat al sistema i el treballador pot consultar les seves dades i el codi QR assignat a aquell usuari
Excepció	**EX1. El sistema falla al guardar usuari** 3. El sistema notifica a hi ha un error al formulari de registre a. L’email del usuari ja esta registrat b. Els camps obligatoris estan buits c. L’email no te el format correcte 4. El sistema retorna al treballador al formulari de registre

Edició d’usuaris

Les empreses podran donar editar usuaris a traves d’un formular, similar al de registre. En aquest formulari, els camps que es podran editar seran els de informació de l’usuari, es a dir, nom, cognom i telèfon.

Nom	Edició d’usuaris
Id	CU_02
Actor	Treballador establiment
Precondició	Treballador ha iniciat la sessió prèviament com a empresa.
Seqüencia Normal	1. El treballador es dirigeix a la pantalla de gestió d’usuaris 2. El sistema mostra un llistat dels usuaris registrats en aquell establiment. 3. El treballador prem el boto de “Donar d’alta usuari” de l’aplicació client. 4. Es mostra per pantalla el formulari de alta. 5. El treballador edita els camps “Nom” i “Cognom”, i prem el boto “Guardar” 6. El sistema guarda les dades de l’usuari. 7. El sistema redirecciona al treballador a la pagina de llistat d’usuaris.
Postcondició	L’usuari s’ha guardat al sistema i el treballador pot consultar les seves dades i el codi QR assignat a aquell usuari
Excepció	**EX1. El sistema falla al guardar usuari** 5. El sistema notifica a hi ha un error al formulari d’edició a. Els camps obligatoris estan buits 6. El sistema retorna al treballador al formulari d’edició, mostrant un missatge d’error.

Donar de baixa usuaris

Les empreses podran donar de baixa usuaris que prèviament havien estat donats de alta, a traves del llistat d'usuaris.

Nom	Donar de baixa usuaris
Id	CU_03
Actor	Treballador establiment
Precondició	Treballador ha iniciat la sessió prèviament com a empresa.
Seqüencia Normal	1. El treballador es dirigeix a la pantalla de gestió d'usuaris 2. El sistema mostra un llistat dels usuaris registrats en aquell establiment. 3. El treballador prem el boto de "Eliminar" sobre l'usuari desitjat. 4. El sistema elimina aquell usuari. 5. El sistema actualitza el llistat d'usuaris de la pagina de gestió.
Postcondició	L'usuari s'ha eliminat i el treballador pot veure la taula d'usuaris actualitzada a la pagina de gestió d'usuaris.

Pagina de gestió, per poder afegir ofertes

A traves del client, es podran gestionar ofertes, ja sigui eliminant les que hi han, o afegint de noves. Aquesta gestió es podrà fer des del programa client, i els camps dels que constaran aquestes ofertes seran nom i descripció, cost en punts, i data d'expiració.
En el futur, es podria augmentar, per personalitzar mes les ofertes, afegint *banners* específics, o imatges de fons.

Per al producte mínim viable que em desenvolupat, la pagina de gestió d'ofertes nomes inclou la funcionalitat d'alta d'ofertes, que es la que ens permetrà poder fer una demo completa. Per tant, nomes inclourem el cas d'us Afegir ofertes

Nom	Afegir ofertes
Id	CU_04
Actor	Manager de l'establiment
Precondició	• Manager ha iniciat la sessió prèviament com a empresa.
Seqüencia Normal	1. El manager prem el boto de "Veure ofertes" de l'aplicació client. 2. El manager prem el boto de "Afegir oferta" de l'aplicació client. 3. L'aplicació client mostra per pantalla el formulari d'alta. 4. El manager emplena els camps "nom", "descripció", "punts" i "data d'expiració", i prem el boto "Afegir" 5. El sistema guarda les dades de l'oferta. 6. El sistema redirecciona al manager a una pantalla de confirmació d'alta.
Postcondició	L'oferta s'ha guardat al sistema i el manager pot consultar les ofertes actives i modificar-les o eliminar-les.
Excepció	**EX1. El sistema falla al guardar la oferta** 7. El sistema notifica a hi ha un error al formulari de alta a. El nom esta buit. b. La descripció esta buida. c. Els punts estan buits o son negatius. d. La data es incorrecta 8. El sistema retorna al treballador al formulari de registre

Generació de codis QR

Per cada usuari registrat, hi haurà associat un codi QR que l'identificarà. Una vegada registrat, el establiment podrà accedir a aquest codi, per imprimir-lo sobre targetes.

Nom	Generació de codis QR
Id	CU_05
Actor	Treballador establiment
Precondició	• Treballador ha iniciat la sessió prèviament com a empresa. • Treballador ha donat d'ata prèviament a un usuari
Seqüencia Normal	1. El treballador prem el boto de "Secció d'usuaris" de l'aplicació client. 2. El sistema mostra un llistat dels usuaris registrats en aquell establiment. 3. El treballador prem sobre el boto "Veure usuari" de l'usuari del que vol veure les dades. 4. El sistema mosta els detalls d'aquell usuari.
Postcondició	L'usuari visualitza els detalls de l'usuari i el codi QR.

Lectura de codis QR i acumulació i subtracció de punts al comprador

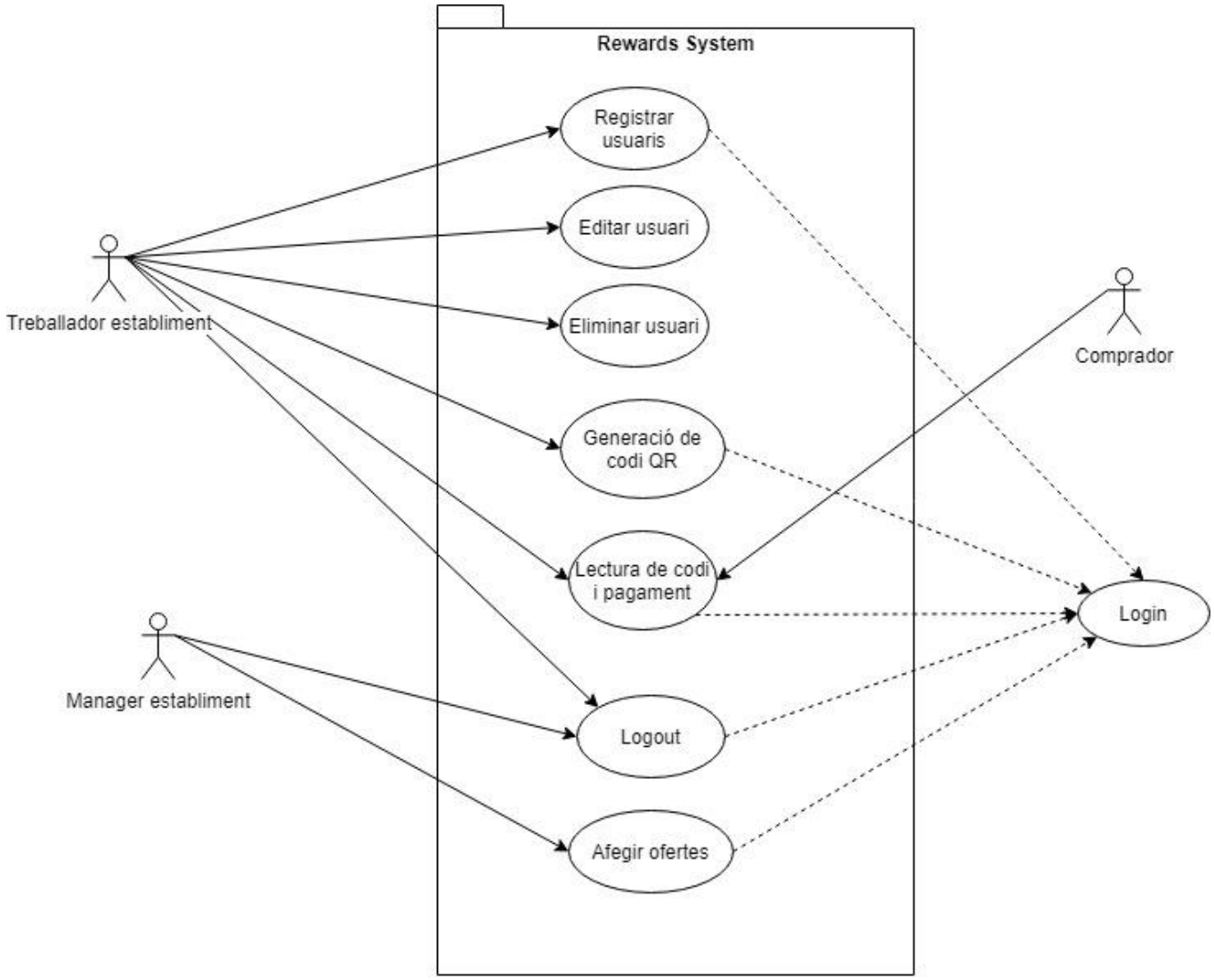
Al fer el pagament, el establiment farà una lectura del codi QR del comprador, per identificar-lo, i passar a fer la acumulació o subtracció de punts. La targeta de l'usuari es llegida per el programa client, instal·lat en un smartphone, i que farà us de la càmera per fer la lectura.

El flux de treball principal serà l'acumulació de punts, on l'usuari presenta la seva targeta, que es llegida per el programa client juntament amb l'import que s'ha pagat. Aquesta informació es transferida a l'API, amb el codi de la targeta, que servirà per identificar al comprador, el codi del client, que serveix per identificar a l'establiment, i l'import, amb el que es calcularà el total de punts. Es farà servir un procés similar en el cas de que l'usuari faci us d'una oferta, i realitzar la subtracció de punts.

Nom	Lectura de codis QR
Id	CU_06
Actor	Treballador establiment, Comprador
Precondició	• Treballador ha iniciat la sessió prèviament com a empresa. • Treballador ha donat d'alta prèviament a un usuari
Seqüencia Normal	1. El treballador es dirigeix a la secció de pagament a traves del boto "Pagament" 2. El treballador selecciona la càmera que farà servir com a lector de targetes, i seguidament prem el boto "Fer servir Càmera". 3. El treballador llegeix la targeta amb la càmera del smartphone. 4. El treballador introdueix l'import i prem "Pagar" 5. L'aplicació client envia l'import i el codi QR llegit.
Postcondició	El sistema converteix l'import en punts i els associa a l'usuari d'aquell QR.
Seqüencia Alternativa	**SA1. El comprador fa us d'una oferta** 1. El treballador introdueix l'oferta que el comprador ha consumit, a traves d'un desplegable. 2. El treballador introdueix l'import i prem "Pagar" 3. L'aplicació client envia l'import, l'oferta consumida i el codi QR llegit. 4. El sistema converteix l'import en punts i els associa a l'usuari d'aquell QR. També resta els punts de l'oferta consumida.
Excepció	**EX1. El sistema dona un error de *time out*** 1. El treballador llegeix la targeta amb la càmera del smartphone. 2. El comprador tarda mes d'un minut a fer el pagament. 3. El sistema mostra un error de temps d'espera sobrepassat.

Diagrama casos d'us

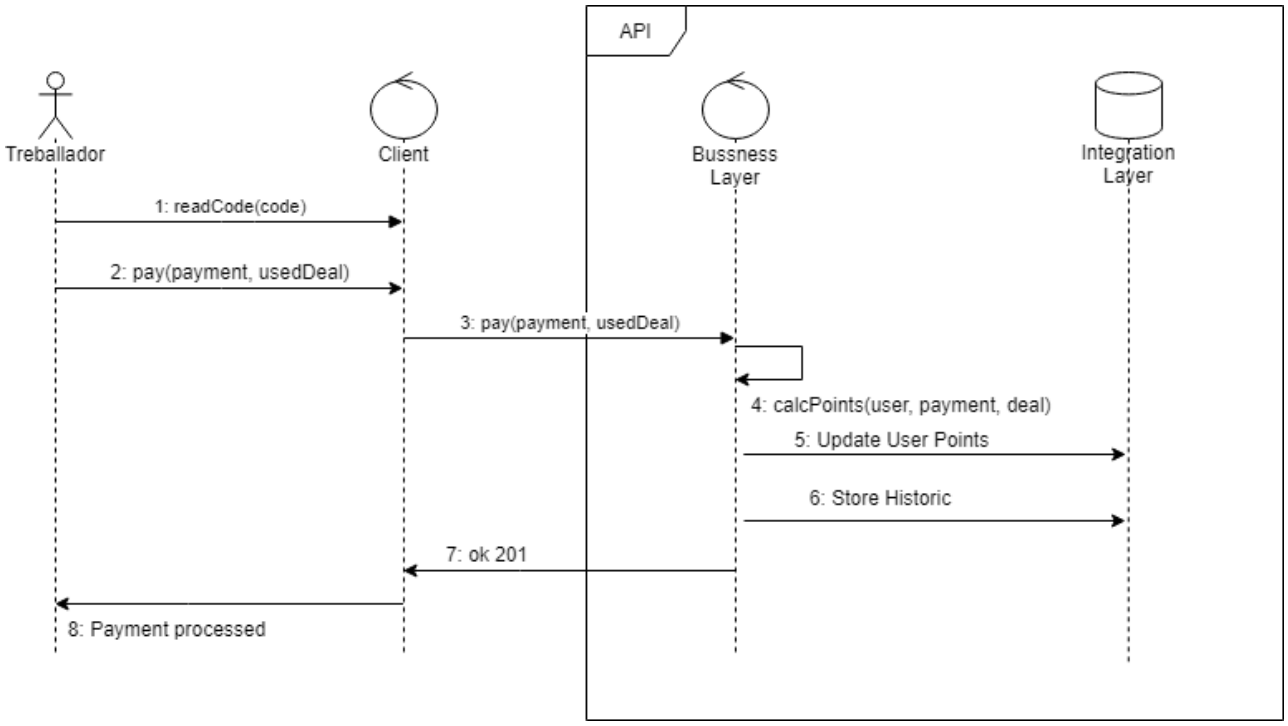
Aquest es el diagrama de tots els casos descrits en l'apartat Casos d'us 2.3:



3. Disseny

3.1 Diagrama de seqüencia

En aquest diagrama de seqüencia, mostrem les diferents interaccions dels objectes en el cas d'us "Lectura de codis QR i acumulació i subtracció de punts al comprador", ja que es el mes complex i rellevant, i que involucra tant el comprador, que es el potencial usuari a qui va destinada la plataforma, com l'usuari de l'organització.

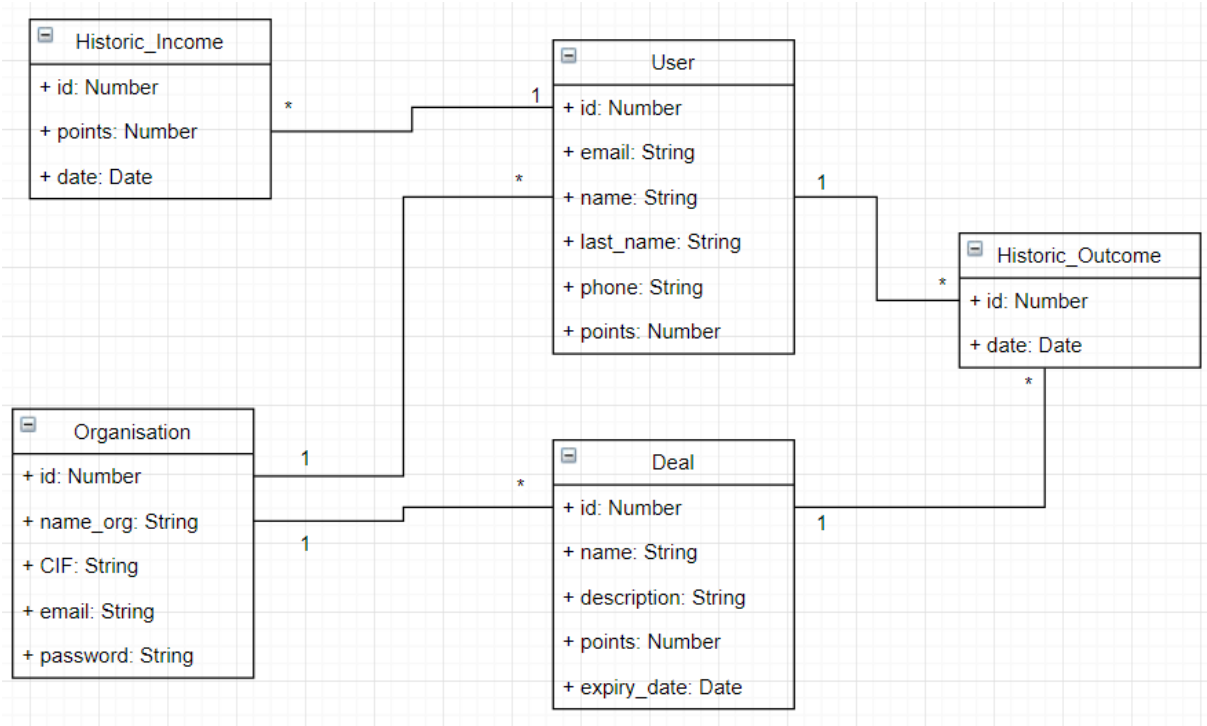


Com podem veure, no hi ha validació de l'usuari en aquest flux. Això es, perquè cada vegada que es client es comuniqui amb la API, aquest inclourà el token JWT a la capçalera d'aquest missatge, de manera que tota acció tindrà una validació d'usuari.

3.2 Disseny relacional de la base de dades

Per a representar el nostre sistema descrit, farem servir un esquema relacional, on tindrem representat l'organització, els usuaris(compradors), i les ofertes. També tindrem dos taules de suport, per portar la comptabilitat dels punts, tant dels acumulats com dels gastats.

El UML inicial per representar el sistema quedaria tal que així:



També adjuntarem els scripts SQL inicials per generar l'esquema:

```
CREATE USER sergi WITH PASSWORD '1234';
CREATE TABLESPACE rewardsspace LOCATION 'C:\Program
Files\PostgreSQL\10\data\dbs';

CREATE DATABASE rewards OWNER sergi TABLESPACE rewardsspace;

DROP TABLE IF EXISTS HISTORIC_INCOME;
DROP TABLE IF EXISTS HISTORIC_OUTCOME;
DROP TABLE IF EXISTS DEAL;
DROP TABLE IF EXISTS BUYER;
DROP TABLE IF EXISTS ORGANISATION;

CREATE TABLE ORGANISATION(
  id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  name VARCHAR (254) UNIQUE NOT NULL,
  CIF VARCHAR (9) UNIQUE NOT NULL,
  email VARCHAR (254) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL
);

CREATE TABLE BUYER(
  id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  email VARCHAR (254) NOT NULL,
  name VARCHAR (50),
  last_name VARCHAR (254),
  phone VARCHAR (9),
  points integer,
  org_id serial references ORGANISATION,
  UNIQUE(email, org_id)
);

CREATE TABLE DEAL(
  id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  name VARCHAR (50) UNIQUE NOT NULL,
  description VARCHAR (500) NOT NULL,
  points integer,
  expiry_date TIMESTAMP NOT NULL,
  org_id serial references ORGANISATION
);

CREATE TABLE HISTORIC_OUTCOME(
  id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  deal_id serial references DEAL,
  user_id serial references BUYER,
  date TIMESTAMP NOT NULL
);
```

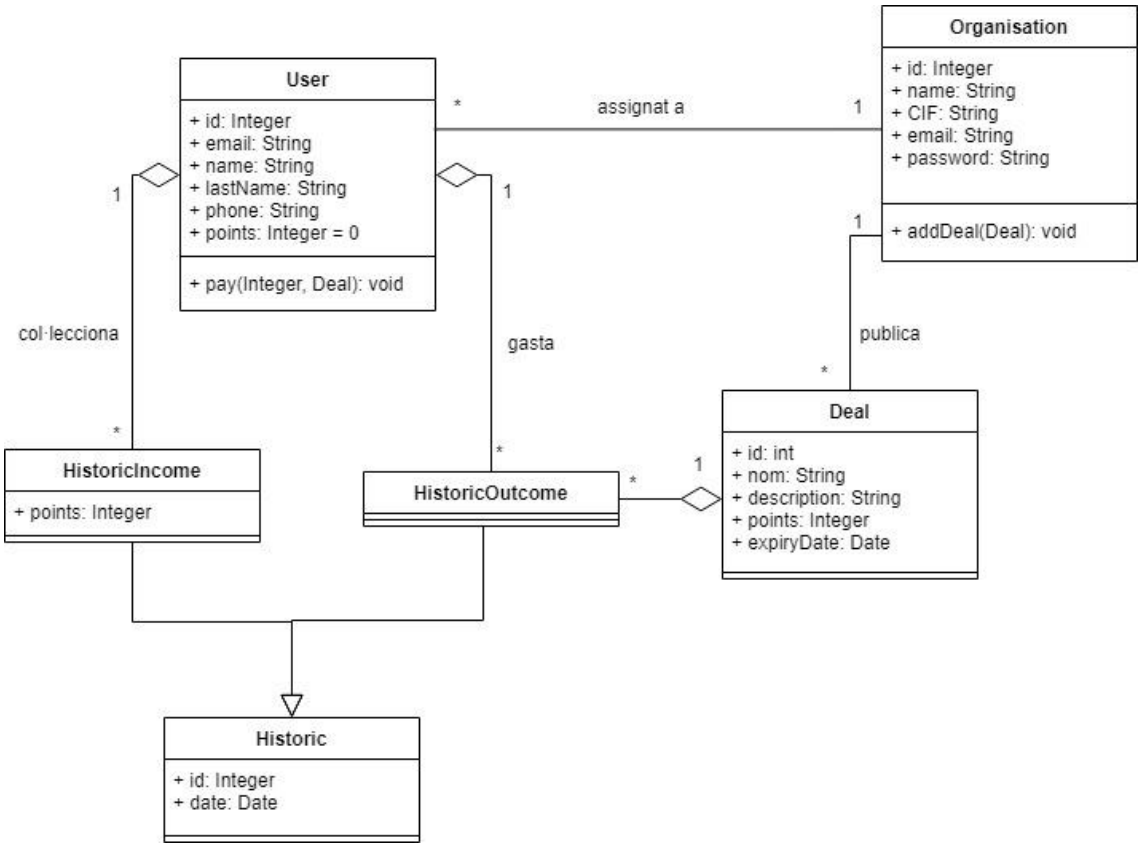
```
CREATE TABLE HISTORIC_INCOME(
  id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  user_id serial references BUYER,
  points integer,
  date TIMESTAMP NOT NULL
);

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO sergi;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO
sergi;
INSERT INTO public.organisation(name, cif, email, password)
VALUES ('RedEye', '113124', 'redeye@redeye.co', '1234');
INSERT INTO public.buyer(email, name, last_name, phone, points, org_id)
VALUES('smontanes@uoc.edu','Sergi','Montanes','973565656',0,1);
```

3.3 Diagrama de classes principals

Una vegada representat el esquema relacional, tindrem que aplicar-lo al nostre projecte JEE, i crear el model per aquest.

Farem us de l'herència, per obtenir els dos Històrics de l'usuari, tant el Income com l'Outcome. També tindrem relacions d'associació dels històrics a l'usuari, ja que son relacions unidireccionals.
Podríem calcular els punts de l'usuari en base als històrics, però s'ha preferit per aquest projecte, el anar calculant cada vegada que fem un pagament o us d'una oferta, i actualitzant els punts de l'usuari, i deixar l'històric únicament a nivell de consulta.



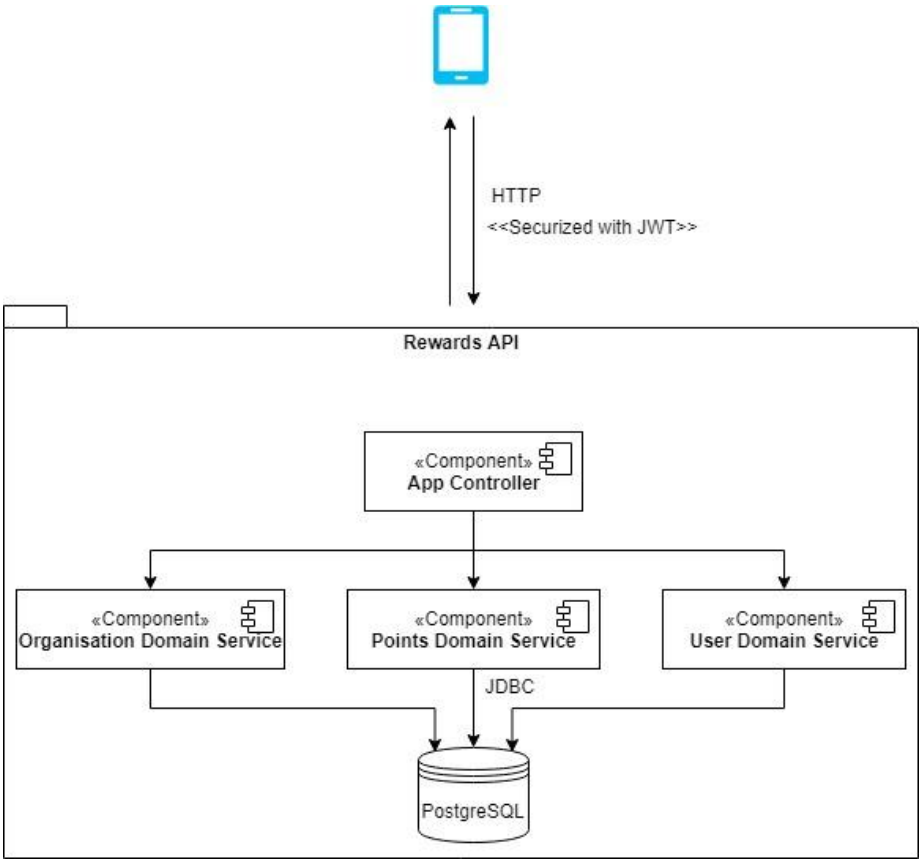
3.4 Disseny de l'arquitectura del sistema

En aquest diagrama, s'aprecia les dos part d'aquest projecte.

- Una aplicació client, que es comunica amb la API de manera *securitzada* amb JWT.
- L'aplicació servidor, que consta d'una API.

Aquestes dos aplicacions conformen una arquitectura client-servidor de 3 capes, sent l'aplicació client, la capa de presentació; el controlador i els serveis la capa de negoci; i el model que representarà tota l'estructura de dades que tenim a PostgreSQL, serà la capa d'integració.

En el cas de la capa de negoci, el controladors serà el punt d'entrada de la API, que avaluarà la petició del client i derivarà la al servei corresponent en cas de ser valida.



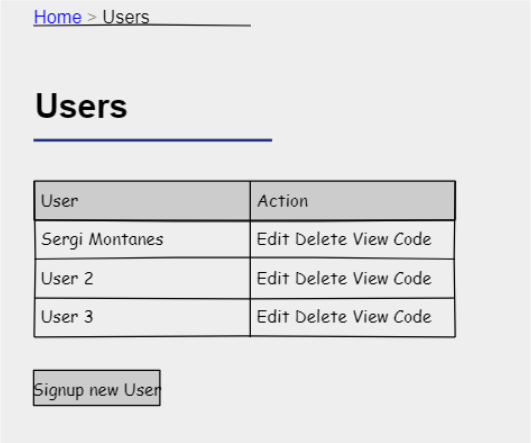
Rewards API està desenvolupat amb Java Spring Boot, usat per a crear micro serveis. Fent servir aquest *framework* ens permet desenvolupar una aplicació autònoma, i la qual ja ve integrada amb un servidor d'aplicacions *Apache Tomcat*.

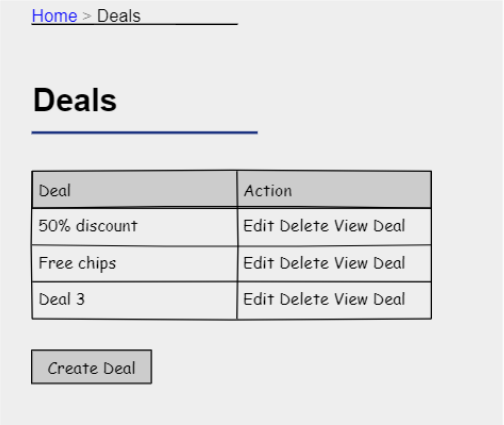
El client estarà desenvolupat amb HTML, javascript i CSS, i connectarà amb la API fent servir l'estil d'arquitectura de programari *REST(Representational State Transfer)* ja que esta pensada per sistemes distribuïts a traves del protocol HTTP, el qual encaixa perfectament en la nostra separació client-servidor.

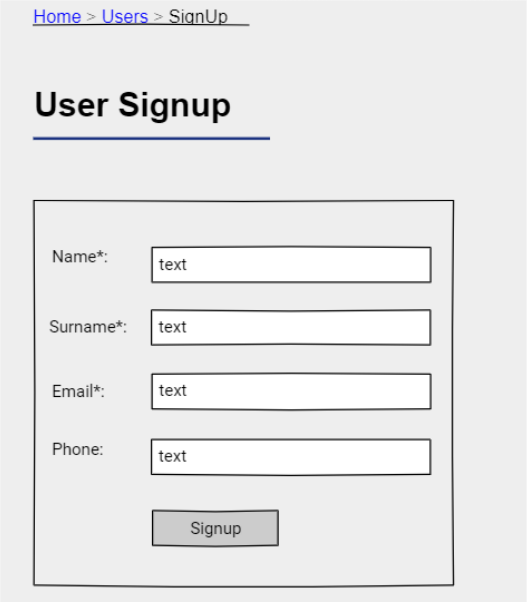
3.5 Disseny de la interfície d'usuari

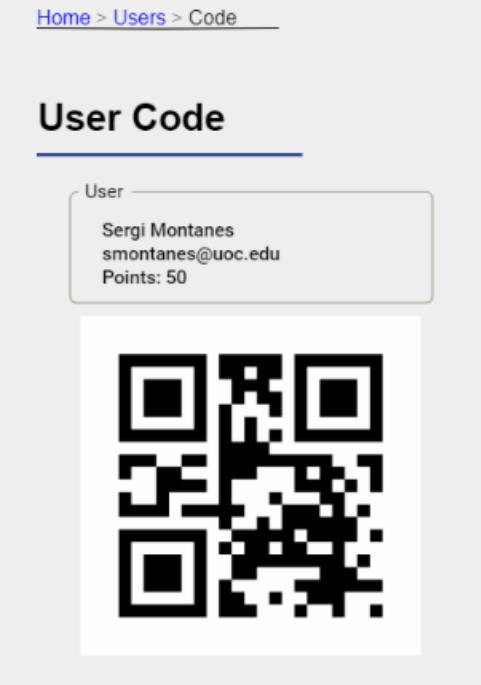
Prototips

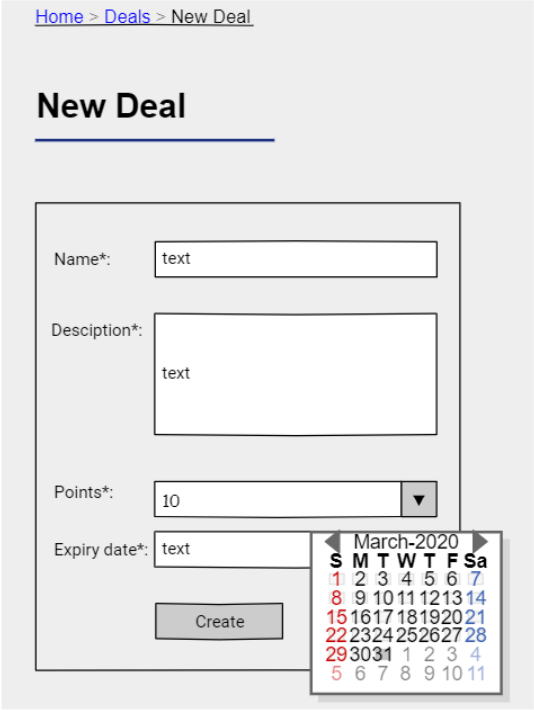
Les següents captures pantalla son les diferents pantalles que es van prototipar, en les fases prèvies a la de desenvolupament.











El disseny inicial era molt clàssic i genèric, i amb amb uns tons poc colorits, tenint marge per a la millora, tal com es pot veure a la secció de versió final.

Versió final

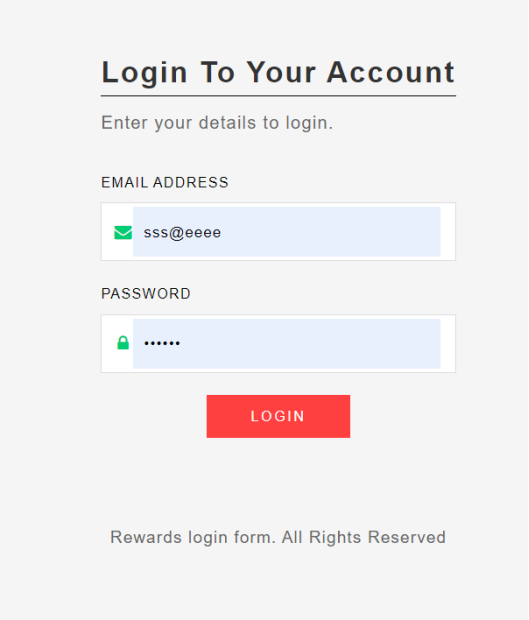
En la versió final, s'ha mantingut l'estil, i el color general, però s'ha ampliat la paleta de colors i millorat l'organització.

L'ampliació de la paleta de colors ens permet ressaltar tots aquells elements de la pagina amb els que l'usuari pot interactuar, com ara hipervincles i botons.

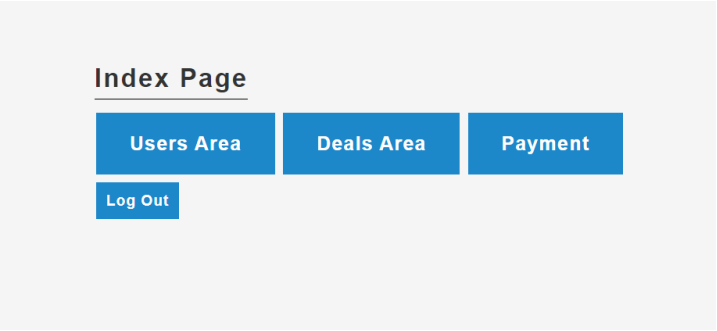
L'organització del menú també s'ha millorat, afegint un nou element que no es trobava en els prototips: la ruta de navegació, també anomenada *breadcrumb,* que ens permetrà moure'ns fàcilment per la nostra web, de secció a secció.

Passem llavors a mostrar cada una de les pantalles, amb una breu explicació.

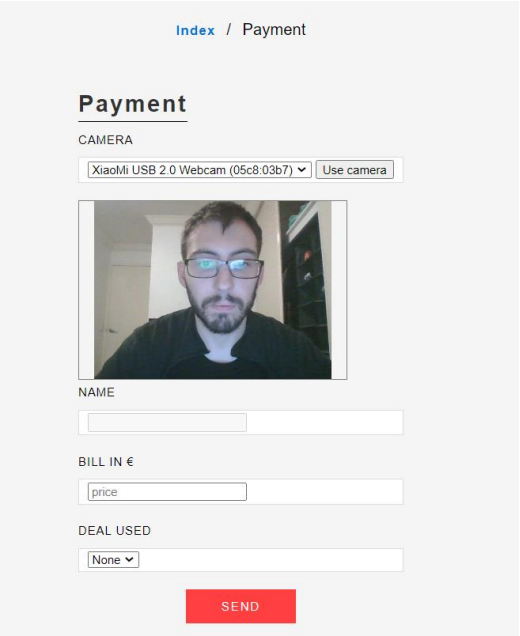
Pantalla de *login*, que permet a les organitzacions donades de alta al nostre sistema, adquirir un *token*, i d'aquesta manera poder accedir a la resta de pantalles i de funcions que la nostra API proveeix.



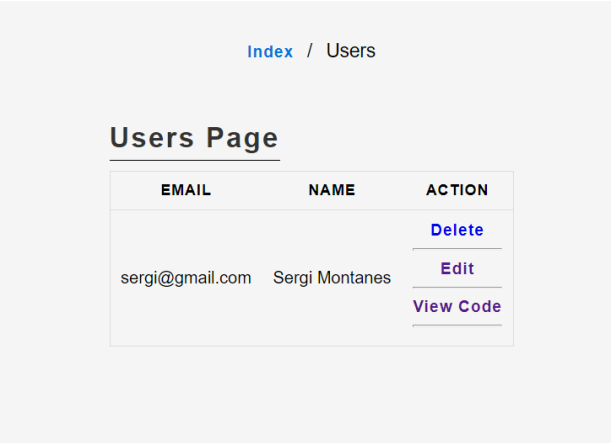
Pantalla índex, la qual dona accés a les tres àrees principals, Usuaris, Ofertes i Pagament. També ens permet desconnectar, el qual ens retornarà a la pantalla de *login.*



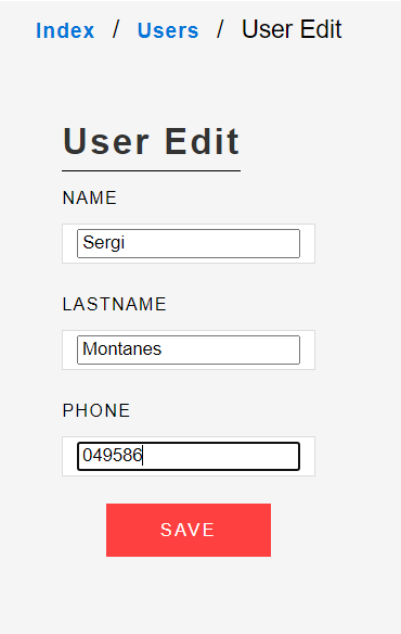
La pagina de pagament, compta amb un visor de càmera incorporat, per poder enfocar i llegir la targeta amb el codi QR. Permet triar quina càmera es vol fer servir, per aquells casos on el dispositiu compti amb mes d'una. Una vegada llegida la targeta, l'usuari associat serà automàticament carregat al client i es mostrarà el seu nom en la caixa de text de nom. També es carregaran les ofertes per aquell usuari, nomes apareixent aquelles que podrà usar amb els punts actuals.



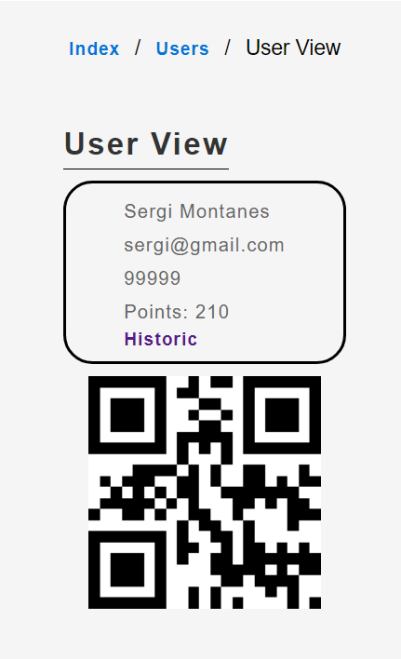
El llistat d'usuaris, ens permetrà veure tots els usuaris associats a la nostra organització, i des de la que podrem executar les diferents accions per a cada usuari.



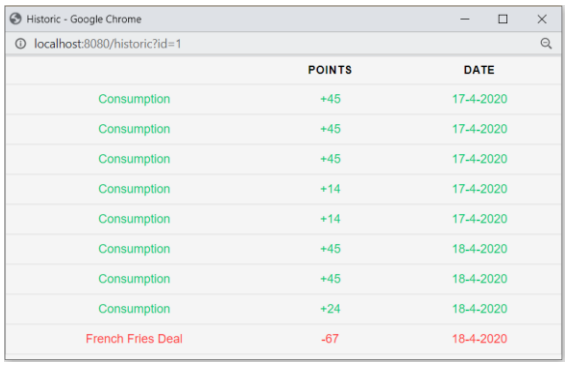
Edició d'usuari, ens permet actualitzar les dades d'usuari.



Informació d'usuari, que ens permetrà veure les dades d'usuari, els punts actuals, el codi QR amb el que s'identificarà als pagaments, i un històric de tots els seus pagaments i ofertes que ha consumit.

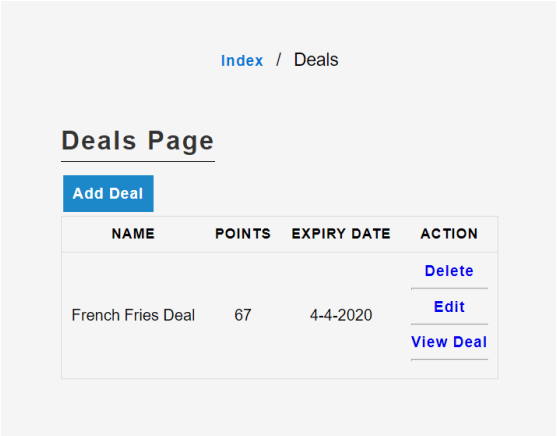


En aquest històric, es podrà veure, el nom de l'entrada(consumició, en cas de que sigui una suma de punts, o el nom de l'oferta en cas de una subtracció), el numero de punts, ja siguin positius o negatius, i la data en que es va realitzar. El color marca si s'han sumat(verd) o restat(vermell) punts

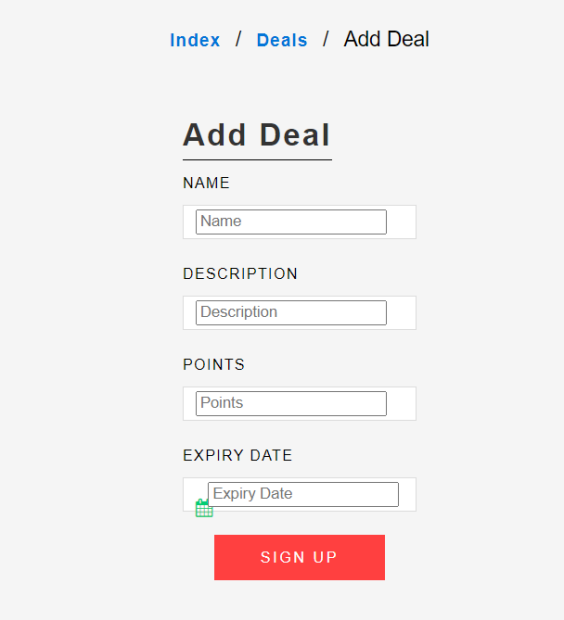


	POINTS	DATE
Consumption	+45	17-4-2020
Consumption	+45	17-4-2020
Consumption	+45	17-4-2020
Consumption	+14	17-4-2020
Consumption	+14	17-4-2020
Consumption	+45	18-4-2020
Consumption	+45	18-4-2020
Consumption	+24	18-4-2020
French Fries Deal	-67	18-4-2020

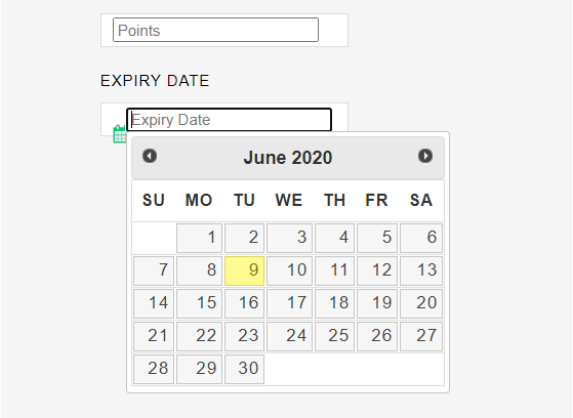
El llistat d'ofertes, que ens permetrà veure tots els ofertes associats a la nostra organització, i des d'on podrem gestionar-les.



La pantalla d'alta d'ofertes ens permetrà donar de alta una oferta amb les seves característiques pertinents, com els punts necessaris per poder aplicar a ella, o la data de caducitat, i associar-la a la nostra organització.



La data de caducitat compta amb un calendari per facilitar la elecció de la data, i fent servir el format correcte.



4. Implementació

4.1 Eines i frameworks

Per la part de la API, hem utilitzat Java EE, amb els següents *frameworks* i tecnologies:

- **Spring**, per poder utilitzar la injecció de dependències, desacoblant l'ús d'un objecte de la seva creació, el que ens ajudarà a seguir els principis d'inversió de dependència i de responsabilitat única.
- **Spring Security**, per poder capturar les peticions provinents del client, ja sigui per generar un token JWT en cas de ser una acció de login, o per validar el token en cas de que sigui una crida a algun altre servei de la *API* .
- **SpringBoot**, que ens permet i facilita crear una aplicació basada en micro serveis, i totalment autònoma(No hi ha necessitat de muntar un servidor d'aplicacions, ja que amb el *framework* ve inclòs un contenidor web, Apache Tomcat).
- **Hibernate**, que ens permetrà mapejar un model de domini orientat a objectes a una base de dades relacional.
- **Gradle**, un sistema de construcció que s'utilitza per a automatitzar la construcció, les proves, el desplegament, etc. Ens facilita el desenvolupament, al permetre'ns automatitzar la construcció del jar, i també la gestió de les dependències del nostre projecte.
- **JWT**. Els JSON Web Tokens és un estàndard obert (RFC-7519) basat en JSON, per a la comunicació de manera segura entre dues parts. En el nostre cas, en serveix per validar l'entitat/organització que s'està comunicant amb la nostra API. Aquesta comunicació s'assegura fent servir una combinació de clau publica - clau privada(RSA).
- **PostgreSQL**, com a sistema de gestió de bases de dades relacionals gratuït i de codi obert.

Per la part del client, hem començat per una web *resposive* amb **HTML5, CSS i JavaScript, amb la llibreria JQuery**, que al ser usada des d'un telèfon mòbil, ens permet fer servir les funcions de la càmera per llegir un codi QR.

També, per la part de lectura i generació de QRs, hem fet servir les llibreries javascript següents:

- **ZXing,** que ens permet la lectura del codi QR i extreure el text que conté, a demes de donar facilitats a l'hora de fer servir les càmeres.
- **QRCode**, que ens permet generar un codi QR a partir d'un text i certs paràmetres per configurar la imatge resultat amb el codi.

En cas d'haver tingut suficient temps, es volia portar aquesta web a *mobile*, fent us de la tecnologia **Ionic**, que ens permetria reutilitzar gran part del codi de la web, i que a demes es compatible amb tots els dispositius mòbils, tant Android com iOS. Aquesta fase es portaria a terme una vegada finalitzada la fase de probes per la API i el client.

4.2 Seguretat

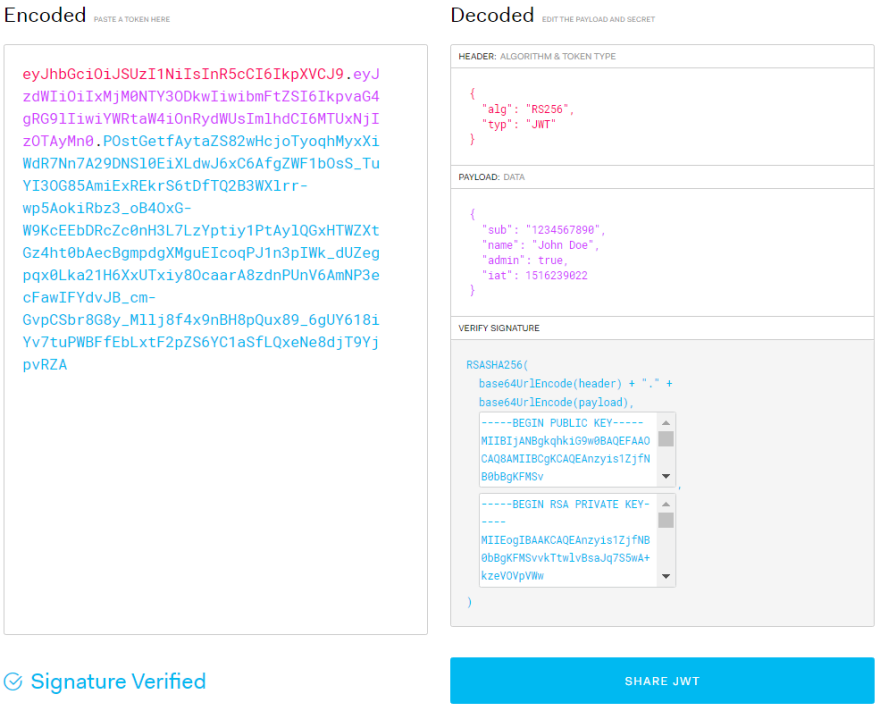
Un dels reptes d'aquest projecte, es el fet de que hi ha una comunicació entre client i servidor, i per tant, tindrem que fer us d'un algoritme criptogràfic, com es JWT, per a securitzar aquest missatges, i protegir-lo contra atacs, com podria ser un *man-in-the-middle, XSS, CSRF...*

Els JWT(JSON Web Tokens) ens permetran transmetre de manera segura informació entre les parts en forma d'un objecte JSON. Aquesta informació pot ser verificada perquè està signada digitalment, usant criptografia asimètrica o de clau publica, que es base en la generació i utilització d'un parell de claus pública/privada.

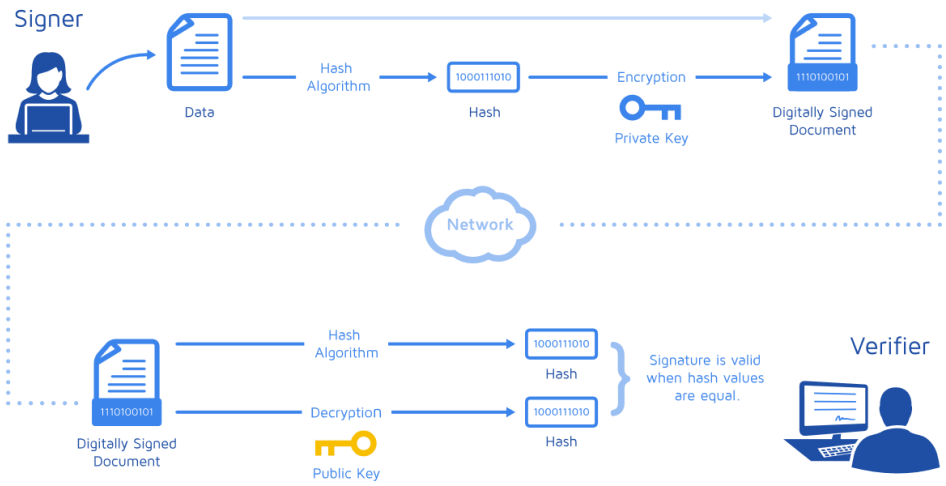
Aquests JSONs tenen una estructura definida i estàndard basada en tres parts: la capçalera, el cos i la signatura.

Mentres que la capçalera i el cos no estan encriptats, sinó que simplement estan codificats en base64 i son públics, la signatura esta encriptada i nomes es podrà verificar amb la clau publica.

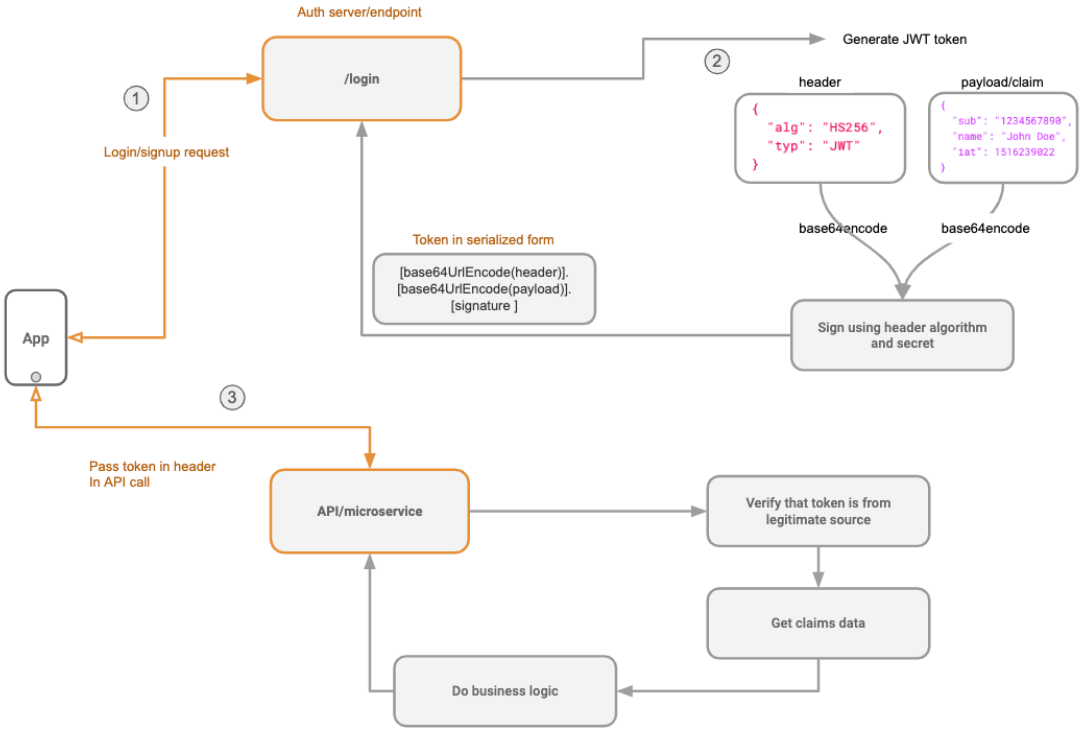
Les primeres dues parts (capçalera i cos) són cadenes de text en base64 creats a partir dos JSON. La tercer part (signatura) es basa en les altres dues parts i les encripta usant un algorisme (en el nostre cas, RSA256). Exemple:



Al ser el signador l'únic que te la clau privada, l'acció de desencriptar-la amb una clau publica permet verificar a l'entitat que l'ha signat.



Per tant, la API de la nostra plataforma serà la que emetrà la signatura per a cada organització que accedeixi a traves del login. El client guardarà aquest token generat, i per cada operació, l'inclourà en la petició. La nostra API doncs, farà servir la clau publica per verificar que aquell token, amb les dades de certa organització, ha sigut generada per nosaltres. Podem veure tot el flux de treball en el següent diagrama:



4. Conclusions

En conclusió, hem pogut presentar un producte mínim viable que es operatiu, i interessant, oferint el servei que es volia, però havent retallat en alguns aspectes de gestió. També una part de documentació que s'ha allargat mes de l'esperat, ja que es podria haver profunditzat mes però degut a la falta de temps, no s'ha fet.

Una de les lliçons apreses d'això, es sempre comptar amb un *buffer* de seguretat, un marge de temps extra a part del temps planificat, ja que com hem comentat, es bastant comú el que s'allargui mes temps del previst, o fins i tot, el tenir que corregir alguns aspectes, com arquitectura, per coses que no s'havien previst.

Tot i així, en aquest projecte, la planificació s'ha seguit casi al peu de la lletra, sobretot centrant-se en les dates en que cada component tenia que estar finalitzat, i a excepció d'algunes funcionalitats de gestió, podem dir que la part tècnica s'ha complert sense massa contratemps. El motiu de les funcionalitats que s'han quedat fora, es sobretot per la falta de familiaritat en certs aspectes de disseny.

Com a programador professional que acostuma a treballar amb Java, la part del servidor ha sigut relativament senzilla i rapida, però coses com el disseny, o la crida dels diferents *endpoints* de l'API, han pres mes temps de l'esperat, el que s'hagués solucionat amb el *buffer* de seguretat mencionat abans. A demes, al ser aquestes funcionalitats molt similar a altres que si que tenim, hagués sigut totalment viable el tenir-les per aquest *MVP*.

En el cas de l'aplicació de mòbil, tot i ser la peça que ens ha quedat fora d'aquest puzle, no estava realment dintre de la planificació, a menys que la part tècnica finalitzes abans del previst.

4. Glossari

- **Ajax**: Sigles de Asynchronous Javascript And Xml, (JavaScript asíncron i XML). Conjunt de tecnologies que permeten actualitzar continguts web sense haver de tornar a carregar la pàgina.

- **API**: Sigles de ‘Application Programming Interface’. Conjunt de regles (codi) i especificacions que les aplicacions poden seguir per a comunicar-se entre elles.

- **App:** Abreviatura de aplicació(del ingles Application)

- **Banners:** Capçalera o anunci que apareix en una pagina web en forma d’imatge.

- **Cloud**: La pràctica d'utilitzar una xarxa de servidors remots allotjats a Internet per a emmagatzemar, administrar i processar dades, en lloc d'un servidor local.

- **CSS:** Cascading Style Sheets (En català: Fulls d'Estil en Cascada). Llenguatge d'estil utilitzat per descriure la semàntica de presentació (l'aspecte i format) d'un document escrit en un llenguatge de marques o HTML.

- **Framework:** Infraestructura de programari que facilita la concepció de les aplicacions mitjançant la utilització de biblioteques de classes o generadors de programes.

- **Javascript:** Llenguatge script basat en objectes, conegut sobretot pel seu ús en pàgines web.

- **JQuery:** Framework de Javascript, creat inicialment que permet simplificar la manera d'interaccionar amb els documents HTML.

- **JWT:** Estàndard obert basat en JSON proposat per a la creació de tokens d'accés que permeten la propagació d'identitat i privilegis.

- **Login**: Procés que controla l'accés individual a un sistema informàtic mitjançant la identificació de l'usuari utilitzant credencials.

- **MVP:** Producte mínim viable (de l'anglès Minimum Viable Product). Producte amb les característiques mínimes per ser emprat.

- **Smartphone**: Telèfon intel·ligent amb capacitats de telèfon mòbil que té un sistema operatiu capaç d'instal·lar aplicacions mòbils.

- **Testing:** Les proves de programari per determinar i validar la qualitat del producte o servei que s'està provant.

- **Token:** Tipus d'estructura de dades que conté una seqüència de caràcters amb un ordre i una llargada determinades.

5. Bibliografia

Articles electrònics:

- Gopal, Tanmai (9 de Setembre, 2019) “The Ultimate Guide to handling JWTs on frontend clients (GraphQL)” https://hasura.io/blog/best-practices-of-using-jwt-with-graphql

- Beatty, Joy (5 de Juliol, 2012) “Visual Models for Software Requirements (Developer Best Practices)” https://seilevel.com/business-analyst-resources/business-requirements-models-templates/use-cases

- Salma (2 de Setembre, 2017) “UML Class Diagrams Tutorial, Step by Step” https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300b

- Selvamanikkam, Meruja (1 de Febrer, 2018) “Digital Signature Generation” https://medium.com/@meruja/digital-signature-generation-75cc63b7e1b4

- Janssen, Thorben - “Map Associations with JPA and Hibernate – The Ultimate Guide” https://thorben-janssen.com/ultimate-guide-association-mappings-jpa-hibernate/

Webs de referencia:

- Spring Framework Oficial doc - https://spring.io/
- PostgreSQL Documentation - https://www.postgresql.org/docs/
- JWT brought by Auth0 - https://jwt.io/
- jQuery API Documentation - https://api.jquery.com/

6. Annexos

Manual execució Rewards System

Part servidora
Per executar la part servidora desenvolupada amb java, farà falta tenir instal·lat al menys la versió 13.0.2 del jre.
Una vegada descomprimit el zip rewards-api.zip, nomes caldrà que anem a la carpeta arrel, rewards-api, i executar les següents comandes amb terminal:

- gradlew clean build
- java -jar build/libs/rewards-api-1.0.0-SNAPSHOT.jar

Amb això, tindrem la part servidora executant-se a un servidor web Apache Tomcat(integrat a la llibreria Springboot), fent us del port 8000.

Part client
La part client, amb la que l'usuari interactuarà. Descomprimirem rewards-fe.zip, i a traves de terminal, ens mourem a la carpeta arrel rewards-fe.
Per executar la part client a un servidor, farem servir node.

https://nodejs.org/en/download/

Una vegada instal·lat, nomes tindre que executar a la terminal:

- npm install --global http-server
- http-server

La part client s'executarà, i serà accessible des de l'adreça localhost:8080

Base de dades
El gestor de Base de dades utilitzat en aquest projecte es PostgrSQL10. Tots els scripts per crear el tablespace, taules, permisos, i un INSERT d'una organització i usuari de proba, es poden trobar a rewards.sql

Repositori online amb el codi font

Tot el codi font esta penjat a GitHub, separat en dos repositoris, un per la part client i l'altra per la part servidora. No es necessari tenir permisos, ja que ambdós son públics.

- **API**: https://github.com/Skeith00/rewards-api

- **Client web**: https://github.com/Skeith00/rewards-fe