

# Poppit

Your tickets have never been this secure

## PTI Project Report

*Poppit: Your tickets have never been that secure*

**Grup 12.2**

*Jordi Soley Masats, Ignasi Juez Guirao,  
Marc Navarro Acosta, Albert Vidal González,  
Edgar Martín Fernandes*

*Tutor: Rubén Tous Liesa*

29 DE MAIG DEL 2023



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Facultat d'Informàtica de Barcelona

**FIB**

## Índex

<b>1</b>	<b>Introducció</b>	<b>3</b>
<b>2</b>	<b>Motivació</b>	<b>3</b>
<b>3</b>	<b>Planificació</b>	<b>4</b>
<b>4</b>	<b>Problemes experimentats</b>	<b>4</b>
<b>5</b>	<b>API</b>	<b>5</b>
5.1	Tecnologia escollida . . . . .	5
5.2	Entitats principals . . . . .	6
5.2.1	Usuaris . . . . .	6
5.2.2	Tickets i Revenda . . . . .	6
5.2.3	Events . . . . .	7
5.2.4	Organitzacions . . . . .	8
5.3	Seguretat . . . . .	8
5.4	Disseny dels endpoints . . . . .	9
5.5	Connexió amb Mongoose . . . . .	9
5.6	Procés de compra, validació i revenda d'una entrada . . . . .	10
<b>6</b>	<b>Front end</b>	<b>11</b>
6.1	Llenguatge de programació . . . . .	11
6.2	Aplicació d'usuaris . . . . .	11
6.2.1	Inici de l'APP - Pàgina principal . . . . .	12
6.2.2	Pàgina esdeveniments . . . . .	12
6.2.3	Pàgina de l'esdeveniment . . . . .	12
6.2.4	Pàgina sel·lecció de tiquets . . . . .	13
6.2.5	Pàgines de registre, inici de sessió, d'usuari i modificació de dades . . . . .	14
6.2.6	Pàgina de tiquets d'un usuari i reventa . . . . .	15
6.3	Aplicació d'organització . . . . .	16
6.3.1	Pàgina d'usuari de la organització . . . . .	16
6.3.2	Pàgina de creació d'esdeveniments . . . . .	16
6.3.3	Pàgina d'esdeveniments . . . . .	16
6.4	Connexions a la API . . . . .	16
6.5	NFC . . . . .	17
6.6	Integració de PayPal . . . . .	18
<b>7</b>	<b>Docker</b>	<b>19</b>
<b>8</b>	<b>Docker Compose</b>	<b>19</b>
<b>9</b>	<b>Jenkins</b>	<b>20</b>
<b>10</b>	<b>Clúster de Kubernetes</b>	<b>21</b>
10.1	Webhook relay . . . . .	21
10.2	Keel . . . . .	22
10.3	Cluster . . . . .	22
10.4	Domini i TLS . . . . .	22
10.5	Persistència imatges . . . . .	23
10.6	Orquestració amb ArgoCD . . . . .	23

---

<b>11 Possibles millores</b>	<b>24</b>
11.1 Afegir mapa events propers . . . . .	24
11.2 Millores NFC . . . . .	24
11.3 Comprar tickets per amics . . . . .	24
11.4 Implementar ofertes a la revenda . . . . .	25
11.5 Multiplataforma (Android, iPhone) . . . . .	25
11.6 Dockerhub premium i fiabilitat del servidors . . . . .	25
11.7 Nom de domini propi i personalitzat . . . . .	25
<b>12 Bibliografia</b>	<b>26</b>

## 1 Introducció

El nostre projecte anomenat Poppit pretén oferir una solució efectiva i funcional per solucionar diferents problemes del món actual de la venda d'entrades per espectacles/events.

Existeixen molts precedents de diferents Events d'interés on de seguida s'han exhaurit les entrades degut al negoci que pot oferir la revenda de les mateixes. Això succeeix perquè existeixen bots i altres eines per automatitzar la venda de les mateixes sense que hi hagi cap tipus de control o regulació d'aquesta activitat.

La solució que proposem es dotarà de la tecnologia NFC per mitigar aquest problema. Volem vincular l'entrada d'un esdeveniment a un identificador únic que existeix en tots els dispositius que contenen un xip NFC. D'aquesta forma, en el moment de verificar si una entrada és legítima es comprovaria si el xip associat al compte que fa la verificació és el mateix que està utilitzant per validar l'entrada. Notem que a banda, també necessitarem limitar el nombre de entrades comprades per Usuari.

També oferirem una funcionalitat més que permetrà revendre l'entrada d'un Usuari que no pot assistir. Tot i això, la revenda formarà part de l'estrategia de negoci de l'Organització propietària de l'Event, ja que aquesta rebrà un percentatge de la venda un cop sigui correctament validada, provocant que tant l'Usuari que ven l'Entrada i la organització obtinguin un benefici de la transacció.

## 2 Motivació

La revenda d'entrades ha incrementat de manera exponencial en els darrers anys. És frustrant veure com les entrades per a esdeveniments populars es venen a preus desorbitats, molt per sobre del seu valor real, mentre que molts aficionats genuïns no tenen accés a elles o han de gastar una fortuna per aconseguir-ne una.

Amb la nostra aplicació, volem posar fi a aquesta situació injusta i garantir que tothom tingui una oportunitat justa d'adquirir les entrades que desitgen per a concerts com els dels Coldplay. La nostra aplicació està dissenyada per connectar directament als fans amb els venedors legítims d'entrades, eliminant els intermediaris i els especuladors que inflen els preus. A més de proporcionar un espai segur per comprar i vendre entrades, la nostra aplicació implementa mecanismes de verificació i autenticació per assegurar que només les entrades vàlides estiguin disponibles en el mercat. Això protegeix als compradors de ser enganyats amb entrades falses o duplicades.

En resum, la nostra motivació per crear aquesta aplicació és proporcionar una solució justa i eficaç als problemes de la revenda d'entrades. Volem crear un entorn segur, transparent i equitatiu on els usuaris puguin gaudir de l'experiència musical que desitgen, sense haver de preocupar-se pels preus exorbitants o les entrades falses. Amb la nostra aplicació, aspirem a canviar la forma en què es compren i venen les entrades, posant els usuaris en primer lloc i fent de l'accés als esdeveniments culturals una experiència més agradable per a tothom.

### 3 Planificació

TASQUES	RESPONSABLE TASCA	COL-LABORACIÓ	DATA INICIAL	DATA FINAL
DEFINICIÓ EN PROFUNDITAT DE LES FUNCIONALITATS	ALBERT VIDAL	TOTS	06/03/23	09/03/23
DISSENYAR ARQUITECTURA I PLANIFICACIÓ DEL FRONTEND	IGNASI JUEZ	JORDI / EDGAR	06/03/23	13/03/23
DISSENYAR ARQUITECTURA I PLANIFICACIÓ DEL BACKEND	MARC NAVARRO	ALBERT	06/03/23	13/03/23
DISSENYAR ENDPOINTS	ALBERT VIDAL	TOTS	06/03/23	09/03/23
CREACIÓ DE LA BASE DADES	ALBERT VIDAL	MARC NAVARRO	13/03/23	15/03/23
DESENVOLUPAMENT DE LA API	MARC NAVARRO	ALBERT VIDAL	13/03/23	14/05/23
DESENVOLUPAMENT DEL CLIENT D'USUARIS FINALS	IGNASI JUEZ	JORDI / IGNASI	13/03/23	14/05/23
DESENVOLUPAMENT DEL CLIENT D'ORGANIZACIONS	EDGAR MARTÍN	JORDI / IGNASI	13/03/23	14/05/23
POSSIBLES MILLORES CLIENTS	EDGAR MARTÍN	JORDI / IGNASI	14/04/23	25/05/23
INTEGRACIÓ DE LA TECNOLOGIA NFC	EDGAR MARTÍN	MARC	14/04/23	11/05/23
INTEGRACIÓ BLOCKCHAIN METAMASK	JORDI SOLEY	ALBERT / IGNASI	14/04/23	11/05/23
INTEGRACIÓ PAGAMENTS	JORDI SOLEY	ALBERT	14/04/23	03/05/23
CI/CD TESTING	MARC NAVARRO	TOTS	15/03/23	19/04/23
DOCKERITZACIÓ DEL SISTEMA	ALBERT VIDAL	MARC	15/03/23	19/04/23
ORQUESTRACIÓ DEL SISTEMA	ALBERT VIDAL	MARC	27/03/23	11/05/23
DOCUMENTACIÓ	JORDI SOLEY	TOTS	06/03/23	25/05/23
PRESENTACIÓ PROJECTE			06/03/23	25/05/23

Figura 1: Diagrama de Gantt final

### 4 Problemes experimentats

Referent a la versió de MongoDB, ens vam trobar amb una limitació en els servidors disponibles. Només es permetia utilitzar la versió 4.4 o anterior de MongoDB, ja que les versions posteriors requerien com a mínim la generació Sandy Bridge de les CPU. Això va resultar en la impossibilitat d'aprofitar les funcionalitats més recents i millors introduïdes en versions més actuals de MongoDB.

En relació als apagaments de les màquines virtuals, vam experimentar problemes de pèrdua de contingut en diverses ocasions. Això significa que, en algun moment, el contingut de les màquines virtuals es va esborrar, implicant la necessitat de recrear-les de nou. Aquesta situació va ser causada per la falta de persistència en les màquines virtuals, les quals no eren capaces de mantenir les dades i configuracions en cas d'apagament o reinici.

Durant el desenvolupament de l'aplicació mòbil, vam enfocar problemes de permisos. En particular, l'ús del framework Flutter va requerir sol·licitar diversos permisos al sistema operatiu del dispositiu mòbil, com ara accés a la càmera, ubicació o contactes. Aquesta sol·licitud de permisos extensos va generar certa resistència o desconfiança en alguns usuaris, ja que havien de concedir una gran quantitat de permisos per tal de poder utilitzar l'aplicació.

Hem de destacar que es va experimentar un accés restringit al servidor a partir de les 12 de la nit durant unes dues hores. Aquesta problemàtica es va poder produir per dur a terme tasques de manteniment i actualització del sistema. Com a conseqüència, durant aquest període horari, l'accés al servidor va ser limitat, afectant a la disponibilitat del desenvolupament i producció del sistema.

En l'àmbit de l'aplicació mòbil, ens vam trobar amb una manca de suport per a la comunicació NFC en el framework Flutter. Concretament, només existia una llibreria disponible per a Flutter que oferia un suport limitat per a la comunicació NFC, i a més, era una versió molt primerenca (versió 0.0.3) basada en Kotlin. Això va limitar la implementació completa de la funcionalitat NFC en l'aplicació, ja que no es disposava d'una solució robusta i totalment compatible amb el framework utilitzat. Aquests problemes van representar desafiaments significatius durant el desenvolupament del projecte i vam haver de buscar solucions alternatives o adaptacions per superar-los. És important tenir en compte aquests aspectes en l'avaluació global del projecte.

## 5 API

### 5.1 Tecnologia escollida

Seleccionar l'anomenat *Stack* tecnològic per al BackEnd és una decisió crucial per al correcte funcionament i èxit del Sistema. Referent a aquesta secció vam avaluar diferents opcions tenint en compte els seus beneficis i buscant l'opció que més encaixés amb l'objectiu del nostre projecte tenint en compte diversos factors, com mida del projecte, complexitat de les estructures, motivació i corba d'aprenentatge.

Després d'una exhaustiva consideració d'opcions on van ser descartades com Flask, Go o Django, vam escollir desenvolupar-la amb Node.js, un popular i eficient entorn de programació que ens ha pogut permetre desenvolupar un Sistema fàcilment escalable i concurrent<sup>1</sup>.

Per a la Base de Dades vam decidir utilitzar MongoDB, una versió NoSQL basada en Documents, que permet gestionar grans volums de dades. Té un esquema flexible i sol anar encapsulada en el mateix *Stack* que Node, cosa que ens va fer acabar de decidir-nos. Aquesta combinació és molt freqüent actualment en les startups. Si bé ha suposat una corba d'aprenentatge en les operacions més complexes com algunes de les que expliquem a continuació, desplegar l'arquitectura bàsica de les entitats no ha estat complicat.

A més, Mongo treballa amb el format JSON i les persones encarregades del Backend hi estaven familiaritzades, a les hores ha sigut més fàcil el tractament de les dades.

Referent al llenguatge utilitzat per Node.js l'API ha estat programda en *Javascript* principalment i també s'ha emprat *TypeScript* per definir correctament els tipatges d'algunes estructures més complexes com podrien arribar a ser els Events.

La nostra aplicació també fa servir imatges i el Backend ha de ser capaç de gestionar-les i de poder treballar amb elles rebent-les i enviant-les a través de peticions HTTP. Per a aquesta funcionalitat s'ha emprat l'*Encoding* amb Base64[8] encara que en un futur es pretén migrar a MIME, per poder soportar millor la càrrega de les imatges sobre les peticions HTTP.

---

<sup>1</sup>Termes que tornarem a mencionar en explicar el funcionament de les peces del Sistema.

## 5.2 Entitats principals

A continuació expliquem les entitats principals en l'arquitectura del nostre Sistema. Per tal d'implementar tènicament aquests conceptes s'ha utilitzat un paradigma de programació orientat a objectes i amb la intenció de millorar l'escalabilitat del Sistema i facilitar la gestió dels arxius i entitats a l'hora de programar, s'ha separat el codi en fitxers[22].

Cada entitat guarda la informació necessària per mantenir la correcta coherència del Sistema, a continuació s'expliquen els camps que guarda cada entitat i que representen.

### 5.2.1 Usuaris

Formen un dels components fonamentals del Sistema, interaccionant amb el mateix per obtenir el Ticket que necessiten. Representen els consumidors que tenen l'objectiu principal d'aconseguir entrades per a Events oferits per la plataforma. Conformen un rol primari en el Sistema i es relacionen amb les altres entitats quan consulten els Events disponibles, consulten les Organitzacions presents al Sistema o executen la compra d'una entrada. Són identificats únicament per un valor associat al hardware del seu dispositiu mòbil a l'hora de validar l'entrada.

Els Usuaris disposen de l'Aplicació d'Usuaris a través de la qual poden consultar informació que el Sistema ofereix així com modificar la seva informació personal.

En la Figura 2 es poden observar els atributs conceptuais implementats per a aquesta entitat a nivell de Sistema.

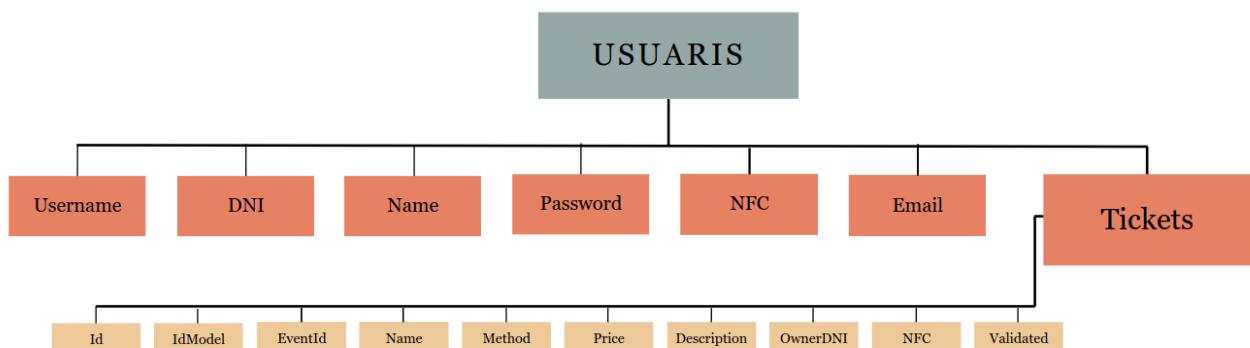


Figura 2: User attributes

### 5.2.2 Tickets i Revenda

Un "Ticket"és un element essencial del nostre sistema que permet l'accés exclusiu a un event específic i està associat de manera única a un usuari. Aquesta funcionalitat és vital per a una de les característiques principals del nostre sistema, ja que garanteix que cada usuari tingui un accés individualitzat als esdeveniments.

Per gestionar adequadament la "Revenda" d'entrades, es va desenvolupar un model que facilita les transaccions entre compradors i venedors. Aquest model permet mantenir la coherència entre les accions realitzades per aquestes entitats en diferents casos d'ús relacionats amb la revenda d'entrades. Això ens permet gestionar de manera eficient i precisa els processos de compra i venda d'entrades, assegurant la integritat i la consistència de les dades involucrades en aquest procés.

### 5.2.3 Events

Aquesta entitat guarda informació sobre un determinat Esdeveniment, que es dona en una data i lloc concret. Gràcies a l'existència d'aquesta Entitat, el Sistema permet consultar tota la informació relacionada, mostrada en la Figura 3, en el moment de compra a través de l'Aplicació.

Un Event pot oferir  $N$  diferents models de ticket, que donen dret a una sèrie d'avantatges dependent del model. La diferència entre models radica en el moment de compra (*depènent de la Release*), el preu i els avantatges que representa el model (*per exemple si dóna accés a la zona VIP*).

D'aquesta forma, tal com es pot observar a la Figura 3, cada Event emmagatzema els seus tickets en forma de matriu. Un Event va vinculat a una organització, formant un node inferior en l'arquitectura. En la figura 4, podem observar els diferents camps que es guarden per cada Ticket venut d'un

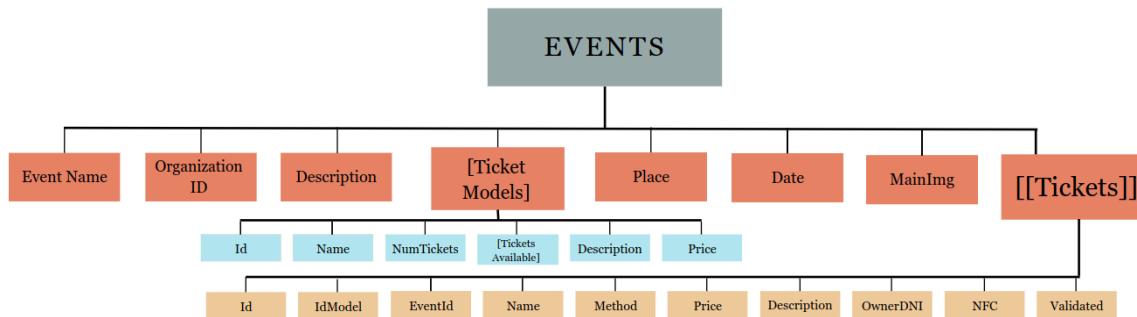


Figura 3: Event attributes

determinat esdeveniment. Notem que aquesta estructura es guarda en forma de matriu, de forma que les files corresponen al model de ticket determinat que s'ha comprat i les columnes corresponen al ticket concret d'un determinat model. Per al ticket en qüestió es guarda tota la informació que podem observar en la Figura 3 il·lustrada amb valors reals a la Figura 4.

	# TICKET 1	# TICKET 2	...	# TICKET N
<b>Standard Model 1st Release</b>	<pre> id: 1 idModel: 1 event_id: 64612881e34dab096f5cb6f2 name: "Standard 1st Release" method: "Original" price: 50 description: "Basic Ticket" ownerDni: 47124856l nfc: 94644cc7-a75f-4bb3-bb75-a8da3969a98 validated: 0 </pre>	<pre> id: 2 idModel: 1 event_id: 64612881e34dab096f5cb6f2 name: "Standard 1st Release" method: "Original" price: 50 description: "Basic Ticket" ownerDni: 47124823o nfc: 526423c7-a75f-44b3-bb75-a8d23d679a98 validated: 1 </pre>	...	<pre> id: N idModel: 1 event_id: 64612881e34dab096f5cb6f2 name: "Standard" method: "REVENUE" price: 60 description: "Basic Ticket" ownerDni: 47124856l nfc: 676fdcc7-a24f-4d3-a75-a0sa3969a02 validated: 0 </pre>
<b>Standard Model 2nd Release</b>	<pre> id: 1 idModel: 2 event_id: 64612881e34dab096f5cb6f2 name: "Standard 2nd Release" method: "Original" price: 70 description: "Basic Ticket" ownerDni: 47124823l nfc: 91244cc7-a75f-4db3-5g75-a8daa3923a10 validated: 0 </pre>			
<b>VIP {N} Model</b>	<pre> id: 1 idModel: N event_id: 64612881e34dab096f5cb6f2 name: "VIP N" method: "Original" price: 200 description: "VIP Ticket" ownerDni: 47124827t nfc: 94644cc7-a75f-4bb3-bb75-a8da3969a98 validated: 0 </pre>	<pre> id: 2 idModel: N event_id: 64612881e34dab096f5cb6f2 name: "VIP N" method: "REVENUE" price: 250 description: "VIP Ticket" ownerDni: 24125756z nfc: 35644cc7-a75f-4bb3-bb75-a8da233969a46 validated: 1 </pre>	...	

Figura 4: Event Tickets

#### 5.2.4 Organitzacions

Una organització representa un grup de responsables que gestionen els Esdeveniments oferits per alguna empresa. Aquests esdeveniments s'ofereixen al gran públic i es poden gestionar a través de l'Aplicació que interactua amb l'API. Aquestes poden oferir nous tipus de ticket per a cada Event, afegir Events o validar els tickets d'un Event.

La creació d'aquesta Entitat ens ha permès guardar informació sobre les Organitzacions<sup>5</sup> per tal que pugui ser accessible per a potencials clients així com fer més fàcil la gestió dels seus Esdeveniments gràcies a l'Aplicació d'Organitzacions.

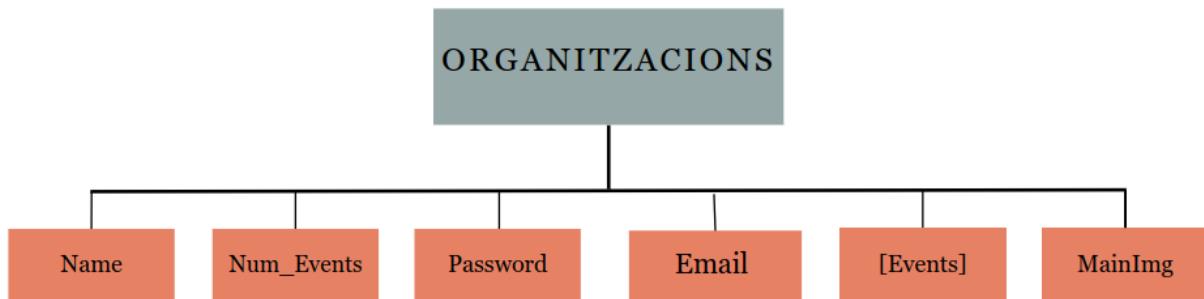


Figura 5: Organization attributes

#### 5.3 Seguretat

En termes de Seguretat s'han avaluat moltes mesures pel fet de ser un factor de vital importància per als desenvolupadors. A continuació s'expliquen algunes mesures que s'han prèss<sup>2</sup>.

1. **Encrypter** és un objecte que utilitza l'API per xifrar i desxifrar[23] els missatges enviats entre la comunicació Aplicació-API. Es situa en un nivell més alt que l'HTTP, és a dir a nivell de protocol d'Aplicació. El missatge quedaria xifrat a nivell d'aplicació i a nivell d'HTTP a sota. S'han generat parells de claus[16] de 2048 bits amb un Sistema de padding *RSA/NONE/OAEPWithSHA1AndMGF1Padding* per tal que l'API i les aplicacions es xifren les peticions entre elles.
2. **Hasher** és un objecte que utilitza l'API per xifrar dades sensibles que s'insereixen a la base de dades. Implementa un Hash **SHA-256** i és utilitat entre d'altres per guardar contrassenyes i fer-ne la validació.
3. **Tokenizer** és un objecte que utilitza l'API per implementar un token JWT[13] per l'autenticació d'usuaris i organitzacions al nostre sistema comporta diversos avantatges. En primer lloc, els tokens s'encripten amb claus úniques per als usuaris i les organitzacions, millorant la seguretat global. En segon lloc, els tokens JWT permeten una autenticació eficient sense consultes constants a la base de dades. En tercer lloc, simplifiquen la gestió de permisos en encapsular la informació rellevant. Finalment, els tokens JWT afavoreixen l'escalabilitat al eliminar la necessitat de gestionar l'estat del servidor. En resum, els tokens JWT ofereixen una millora en

<sup>2</sup>Notem que en el moment de treure l'Aplicació a producció s'hauria de ajustar alguns detalls a nivell de codi.

la seguretat, una autenticació eficient, una gestió simplificada de permisos i una escalabilitat millorada al nostre sistema.

4. La implementació de **HTTPs** en el nostre sistema té com a objectiu encriptar les peticions a nivell d'aplicació, proporcionant una capa de seguretat addicional durant la comunicació entre l'API i els clients. A través de HTTPS, les condicions de xifrat es configuren de manera acordada en cada connexió, garantint la confidencialitat i integritat de les dades transferides. Aquesta implementació reforça la seguretat global del sistema, protegint les comunicacions contra possibles interceptacions o manipulacions no autoritzades.

## 5.4 Disseny dels endpoints

El disseny de l'arquitectura que seguirà els endpoints és molt important perquè defineix com seràn accedides les funcionalitats principals del programa.

Endpoint	Explicació
/user	Endpoints relacionats amb l'Usuari i la gestió de les seves compres.
/events	Endpoints que permeten accedir a la informació i accions relacionades.
/organization	Endpoints que serveixen per modificar informació i Events de les organitzacions.
/getImage	Endpoint que serveix per solicitar alguna imatges guardades per la API.

## 5.5 Connexió amb Mongoose

Per tal que la API pugui oferir un servei correcte als Clients i assegurar la persistència de les dades en diferents escenaris com fallades, manteniment o simplement connexió/desconnexió dels Clients, és crucial garantir que el servei estigui sempre disponible juntament amb la informació dels Usuaris, Esdeveniments i Organitzacions.

Per resoldre aquest problema, hem optat per utilitzar MongoDB, tal com s'ha detallat en l'apartat "Tecnologia escollida" del nostre projecte. MongoDB és una base de dades No Relacional que treballa amb documents i permet guardar i obtenir resultats d'un conjunt de dades en format JSON. Per assegurar una organització adequada, hem configurat una estructura de Document per a cada entitat del nostre sistema.

Per facilitar la comunicació entre l'API desenvolupada amb el framework Express i la base de dades MongoDB, hem aprofitat el mòdul de Node.js anomenat Mongoose. Hem implementat una classe específica, anomenada "database\_handler", que és responsable de totes les operacions relacionades amb el guardat, recuperació, modificació i eliminació de dades a la base de dades.

Cada entitat del nostre sistema té un Document associat en la base de dades, on es guarden les seves dades i es gestionen per mantenir la coherència. Això ens permet treballar de manera estructurada i organitzada amb les dades dels Usuaris, Esdeveniments i Organitzacions, assegurant que l'API ofereixi un servei robust i fiable.

Gràcies a l'ús de Mongoose i a la configuració adequada dels Documents per a cada entitat, hem aconseguit una integració eficient entre l'API i la base de dades MongoDB, garantint la persistència de les dades i assegurant que el sistema sigui capaç de gestionar els diferents escenaris que es puguin presentar, sense comprometre la disponibilitat del servei ni la integritat de la informació.

## 5.6 Procés de compra, validació i revenda d'una entrada

En aquest apartat es detallarà el procés i el *management* de les estructures de dades que es validen i s'editen quan hi ha una compra<sup>3</sup>.

Cal distingir que el procés de compra es segmenta principalment en dues fases: la reserva i la confirmació. En la primera fase, es consulta que quedin tickets disponibles del model que s'està cercant. Això ho podem esbrinar gràcies a les estructures *num\_tickets* que indica el nombre de tickets que té un determinat model *m* i l'estructura *tickets\_available* que és un array amb l'identificador de tots els tickets lliures del model *m*.

1. En cas de que *tickets\_available* != [], es treu un identificador de l'array i es retorna com a resposta de la petició.
2. En cas de que correspongui un array buit, es retorna error.

Un cop aquesta fase de reserva ha estat superada, l'Aplicació del Frontend verifica que el pagament sigui correcte i un cop dona l'aprovació es procedeix a la confirmació, on s'adjunta l'identificador de ticket retornat en l'anterior petició juntament amb el model de ticket.

Quan es verifica que el model i l'identificador del ticket són correctes, el Sistema cerca l'Usuari i li assigna el ticket amb l'identificador *NFC* únic de l'Usuari. I també li insereix a la seva entrada del Document *Usuaris* la informació corresponent al Ticket amb l'etiqueta *validated* a 0.

En el punt actual del cas d'ús l'Usuari opta principalment a dos escenaris: anar a l'Esdeveniment validant la seva entrada o posar l'entrada a la venda a través de l'Aplicació, per intentar recuperar una part del seu cost un cop aparegui el *sold out*.

En el primer cas, l'Usuari tindria l'entrada a la seva informació i gràcies a les tècniques NFC explicades en apartats posteriors podríem comprovar si la informació que s'està transmetent concorda amb la que tenim de l'Usuari, és a dir, verificar que sigui ell qui accedeix a l'Event llegint el tag NFC. Per verificar l'entrada s'enviaria una informació mitjançant la tecnologia NFC a la API, i si aquesta concorda, posaria el bit de *validated* a 1, donant accés a l'Event(*return 200 de l'API al frontend*) i invalidant les possibles accions posteriors relacionades amb aquell ticket, com intentar una revenda d'un ticket validat.

En el segon cas, l'Usuari proposaria un preu, mai inferior a l'original per publicar l'entrada a la revenda oficial de l'Organització. Un cop l'API rep aquesta petició fa comprovacions de seguretat, com per exemple que l'entrada no estigui validada, que pertany a un Usuari o que l'Esdeveniment no hagi succeït. Si tot això es compleix l'entrada es guardaria a un document temporal *Revenues*, que guarda informació de totes les entrades de Revenda que seràn accessibles quan l'Event faci *sold out* dels tickets oficials.

---

<sup>3</sup>Notem que l'Usuari ha d'haver iniciat sessió per poder optar a comprar un ticket

## 6 Front end

### 6.1 Llenguatge de programació

Per a desenvolupar la nostra aplicació vam haver de fer una recerca ràpida sobre quines plataformes ens permetrien arribar al nostre objectiu de desenvolupar una aplicació des de zero. Les dues opcions que varem tenir en compte van ser AndroidStudio i Flutter. En primera instància vam decidir utilitzar Flutter, aquesta decisió es va basar en el fet que aquest ofereix un inici més acollidor per als nous usuaris en termes de programació i disseny de l'aplicació. No obstant això, al llarg del desenvolupament de l'aplicació hem experimentat diversos problemes a causa de la manca de suport i la manca de documentació en alguns recursos degut a lo recents que eren, per a desenvolupar les funcionalitats de la nostra aplicació. [27]

Una de les característiques més importants que volem destacar de Flutter és el “hot reload”, aquesta funcionalitat ens permet veure en temps real els canvis que es van generant en la nostra aplicació en termes de disseny i funcionalitats. Això fa que el procés de desenvolupament sigui més ràpid i eficient ja que els canvis es veuen de forma immediata.



Un dels problemes principals que ens hem enfrontat ha sigut la falta de desenvolupament de paquets i llibreries. Aquests eren massa recents i no disposaven d'una documentació per a poder utilitzar de la forma adequada o directament el seu funcionament no era l'adequat. Això ens ha suposat un sobretemps extra per trobar alternatives a les funcionalitats a desenvolupar en el projecte.

A causa d'aquests problemes i dificultats experimentades durant el nostre projecte, si en un futur féssim un altre projecte, consideraríem les alternatives a Flutter. Avaluariem les necessitats específiques del projecte i considerariem l'opció de utilitzar una tecnologia que ofereixi un suport més complet per a les funcionalitats que es vulguin implementar.

És important recordar que cada projecte té les seves pròpies necessitats i les decisions han d'anar acord amb aquestes. Tot i que Flutter pot ser una opció viable per a molts desenvolupadors, la nostra experiència personal ens ha portat a considerar altres alternatives en futurs projectes.[26]

### 6.2 Aplicació d'usuaris

L'aplicació d'usuaris està destinada a tots aquells que són amants de la música, l'esport i l'entreteniment en general, ja que sabem com és important per a l'usuari poder accedir fàcilment als millors esdeveniments de diferents localitats. L'aplicació disposa d'una àmplia selecció d'esdeveniments degut a una àmplia base de dades, des de concerts en directe fins a partits d'esports i festivals, l'aplicació connectarà a l'usuari amb una àmplia i variada gamma d'opcions.

Aquesta aplicació consta d'un login i un registre per a tots els usuaris finals, una pantalla principal on es mostraran tots els esdeveniments disponibles i les seves respectives pantalles per a la seva compra, una pantalla per a la informació de l'usuari i finalment una pantalla de fàcil accés per a l'usuari per a la compra/venda de tiquets de reventa.

### 6.2.1 Inici de l'APP - Pàgina principal

Primerament quan l'aplicació s'inicia es mostra la pantalla principal amb una secció de esdeveniments propers temporalment, una d'organitzacions i el mapa per esdeveniments propers en quan a localització (aquest últim no està implementat, no es funcional, però podria ser una de les seccions que es podria dur a terme en un cas real).[5] [25] [24]

### 6.2.2 Pàgina esdeveniments

Aquesta pàgina permet veure tots els esdeveniments creats per totes les organitzacions que participen en la app. Per cada esdeveniment es mostra la seva informació principal o rellevant i la imatge destinada a ell. A partir d'aquí es poden premer tots els events per accedir a la seva informació, com a la compra d'entrades per al event triat.

### 6.2.3 Pàgina de l'esdeveniment

Aquesta pàgina disposa de tota la informació d'un esdeveniment; fotografia identificadora de l'organitzador de l'esdeveniment, nom del esdeveniment, localització i hora, informació sobre el tipus de tiquets disponibles a la venda i un accés per a la seva compra.

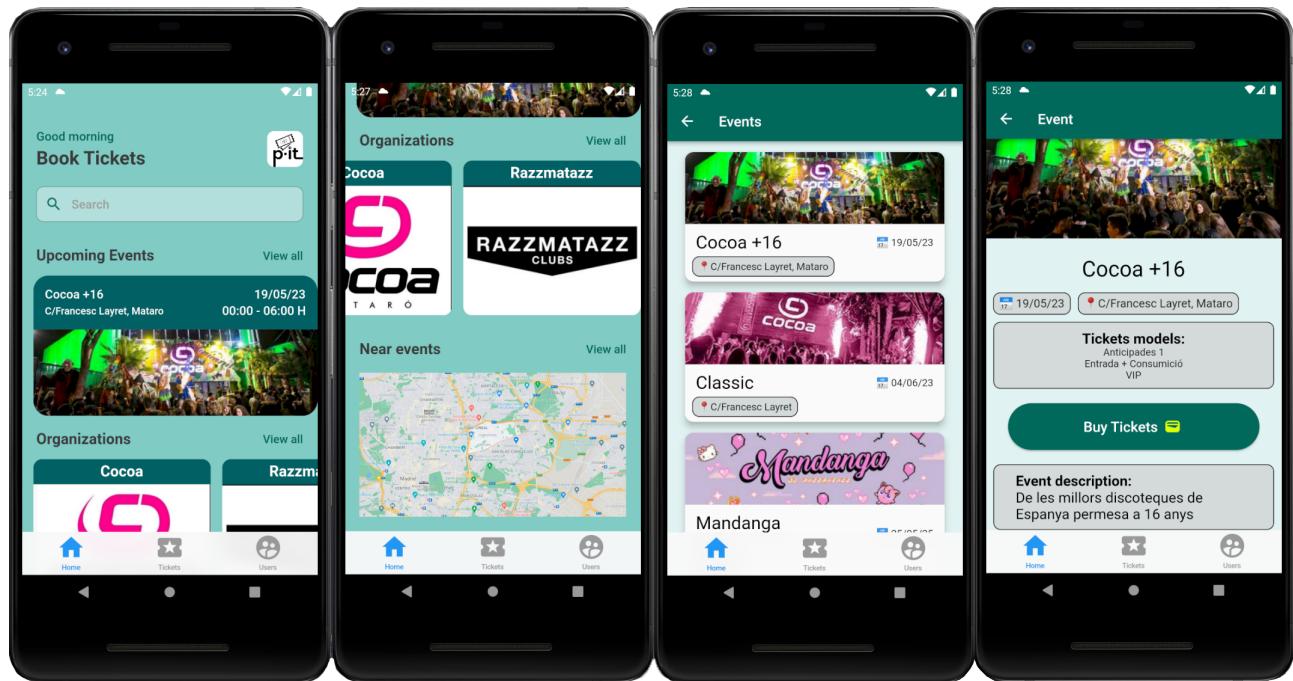


Figura 6: Menú principal, llistat d'esdeveniments i informació d'un esdeveniment

#### 6.2.4 Pàgina sel·lecció de tiquets

Quan un usuari accedeix a la pàgina de sel·lecció de Tiquets, no cal que estigui loguejat.

En total com a màxim es podràn sel·leccioñar 6 unitats, on per cada model de Tiquet es podràn sel·leccioñar →  $\text{MIN}\{6, \#TIQUETS DISPONIBLES D'AQUEST TIPUS\}$ . Un cop sel·leccioñades les unitats màximes permeses, l'ícone de l'esquerra inicrà alerta, de no poder sel·leccioñar més unitats.

En el cas que un tipus de Tiquet, no tingui unitats disponibles, s'enfosquirà amb un missatge de **Soldout**. L'ícone de l'esquerra indicrà alerta, de no poder sel·leccioñar unitats.

El botó inferior s'anirà actualitzant amb el preu segons es sel·leccioñen els Tiquets i si no n'hi ha disponibles, al fer clic et portarà a la pantalla de compra de Tiquets de reventa.

- En cas de no estar loguejat, t'avisiñarà al fer clic.

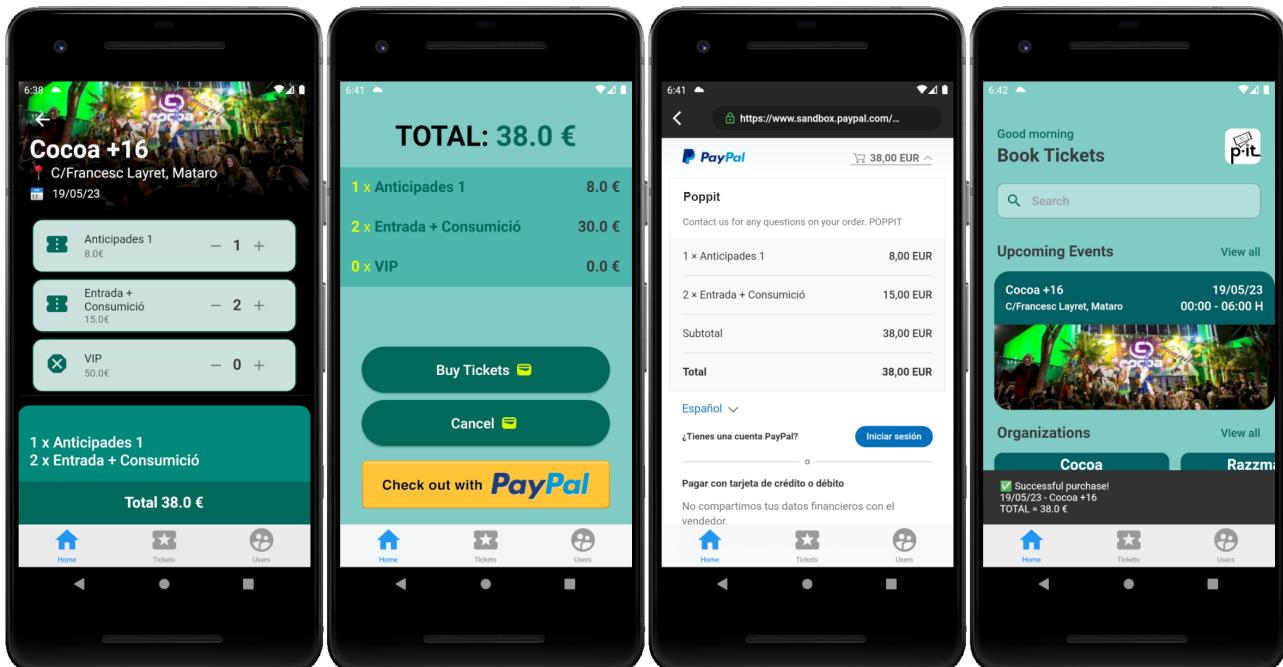


Figura 7: Procés de compra de tiquets

### 6.2.5 Pàgines de registre, inici de sessió, d'usuari i modificació de dades

La pàgina de registre permet a l'usuari que no té un compte poder fer-se un, on tots els camps s'han d'omplir per a poder realitzar un registre exitós. La contrasenya s'ha d'introduir dues vegades per a major seguretat i idèntiques, altrament s'informarà amb un error a l'usuari.

La pàgina de login permet a l'usuari fer login amb les seves respectives credencials, en cas que no estigui loguejat té l'opció per anar a fer el seu registre en l'aplicació. Donarà error en cas que les credencials siguin incorrectes i no es realitzarà el login mostrant un missatge per a informar a l'usuari.<sup>[4]</sup>

La pàgina del perfil d'un usuari el permet veure les seves credencials actuals junt amb les seves dades més rellevants. Des d'aquí també es pot accedir a les opcions de tancar sessió per a poder inicar-ne de nou amb un nou usuari, i poder modificar els atributs i valors de l'usuari, així com la seva contrassenya.

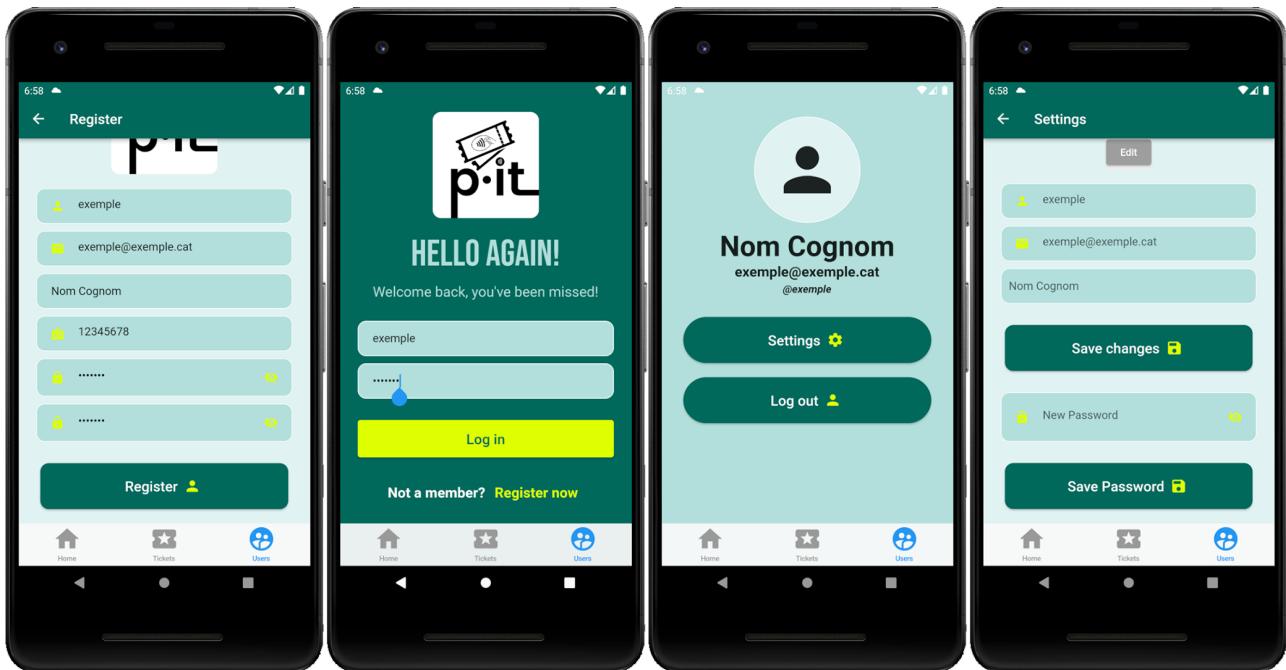


Figura 8: Registre, inici de sessió, perfil i canvis

### 6.2.6 Pàgina de tiquets d'un usuari i reventa

La pàgina de tiquets només està disponible si l'usuari estàloguejat, ja que mostra tots els tiquets dels quals n'és propietari. Des de aquesta pàgina es pot accedir a validar qualsevol dels tiquets que es tingui en propietat amb la funcionalitat del NFC o posar-los a la venda de nou un cop s'hagin esgotat els tiquets per a l'esdeveniment.

La pàgina de reventa permet a l'usuariloguejat poder vendre un tiquet, disposant d'un camp per a posar valor al seu ticket. Al posar el valor al ticket i posar-lo a la venda, el ticket romandrà al compte de l'usuari fins que aquest sigui venut. Aquesta pàgina només serà útil si els tiquets de l'esdeveniment del qual pertany el ticket, s'han esgotat previament.

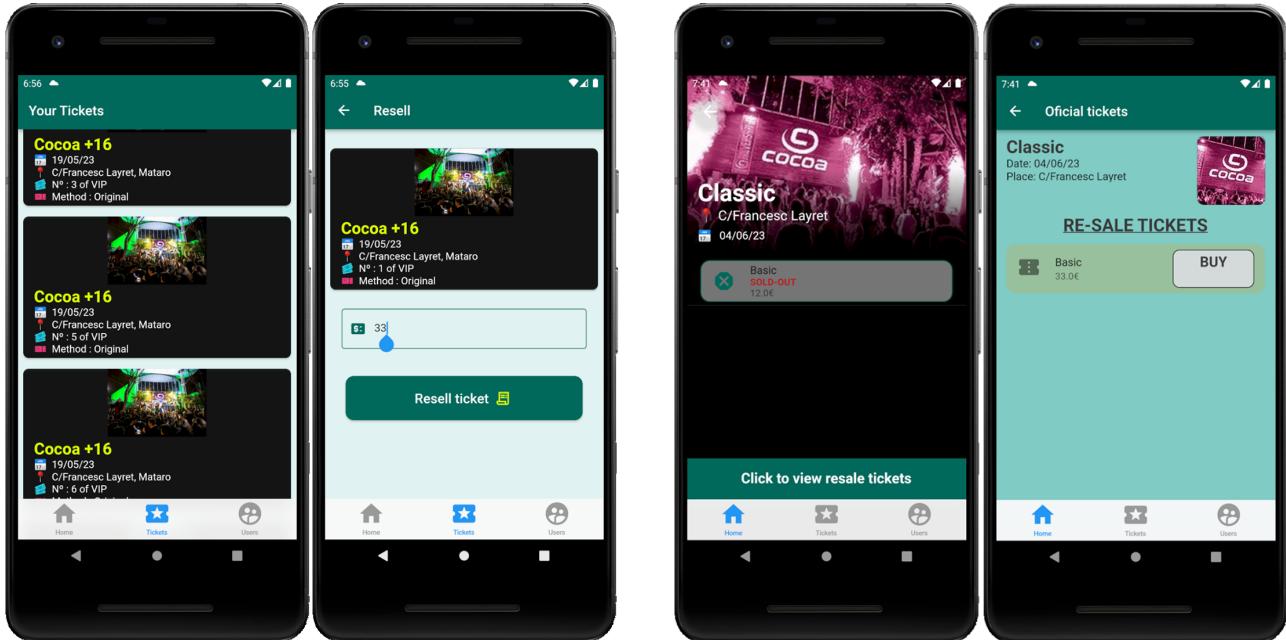


Figura 9: Tiquets d'un usuari i reventa

### 6.3 Aplicació d'organització

La APP d'organitzacions serà la donada a les empreses amb un usuari i contrassenya donats per la empresa de poppit al voler utilitzar la nostra tecnologia. Aquesta aplicació contsa d'un login amb les credencials donades per nosaltres, i al validar-les amb una crida a la api, s'entra al menu de l'aplciació amb tres pantalles principals.[18]

#### 6.3.1 Pàgina d'usuari de la organització

La tercera opció i la més simple és la pàgina d'informació de la organització, on al clicar-la sinplement es desplega la informació de la organització donada a nosaltres al voler-se apuntar al nostre projecte. Aquesta informació no permetem que pugui ser modificada directament, ja que es informació important i preferim que l'empresa no tingui control directe i s'ens notifiqui del canvi per així poder-ho canviar nosaltres.

#### 6.3.2 Pàgina de creació d'esdeveniments

La segona pàgina consisteix en un formulari que permet a la organització omplir-lo amb tota la informació d'un esdeveniments, la seva imatge, i els models de tiquets que s'ofereixen.

#### 6.3.3 Pàgina d'esdeveniments

Aquesta podria dir-se que és la pàgina més important de l'aplicació ja que permet veure tots els esdeveniments creats per l'organització i interactuar amb ells. Al premer un dels esdeveniments creats per l'organització, apareix la seva informació i la aplicació permet modificar les seves dades, eliminar i afegir models de tiquets, eliminar l'esdeveniment o validar-lo a través del NFC.

### 6.4 Connexions a la API

Per connectar-nos al Cloud de la FIB utilitzem crides a la api exposada al port 40341. Els missatges enviats amb post i get utilitzen encriptació asimètrica ja que s'encripten amb la clau publica de la API i després la propia API utilitza la seva clau privada per desencriptar el missatge. A més utilitzem https per augmentar la seguretat. Amb les crides a la API s'envien i reben json's i imatges codificades en base 64. També al fer login un usuari, se li assigna un token al usuari per a les posteriors crides a la API on comprova el token i només permet crides amb un token vàlid.

## 6.5 NFC

La finalitat de la connexió NFC és aconseguir que dos mòbils parlin entre ells mantenint les entrades en estat digital tota l'estona per a que no es puguin copiar com els codis QR o revendre a preus desorbitats.

Per això la nostra idea inicial era utilitzar un sistema peer-to-peer a través del NFC, cosa que ja no és possible per temes de seguretat en android i iOS, deixant-nos així amb la unica possibilitat de fer-ho amb el sistema read/write que és l'unic suportat avui dia. Al ser read/write, no es poden escriure ni llegir capçaleres de dades NDEF que són les que estan dins dels NFCTags, ja que els mòbils no emmagatzemem les dades en el seu xip NFC d'aquesta forma. Així que la solució era fer que el mòbil que tingués oberta l'aplicació d'usuari, emulés una targeta NFC com per exemple la del banc o la T-mobilitat.

El problema de fer servir aquest mètode és que Flutter és molt recent i les uniques llibreries amb les quals es podria fer el HCE (Host Card Emulator)[1] són massa noves i en una versió Alpha, lo que comporta que no funcioni bé.

L'alternativa a això és fer la llibreria per nosaltres mateixos, però degut a la manca de temps total que es té per fer el projecte, no es viable. Per això la única possibilitat viable és utilitzar un intermediari que en aquest cas és un tag NFC el qual permet llegir i escriure desde diferents mòbils. Per tant el sistema de validació consisteix en que el mòbil del client escriu al tag i el de la organització valida la informació escrita al tag.

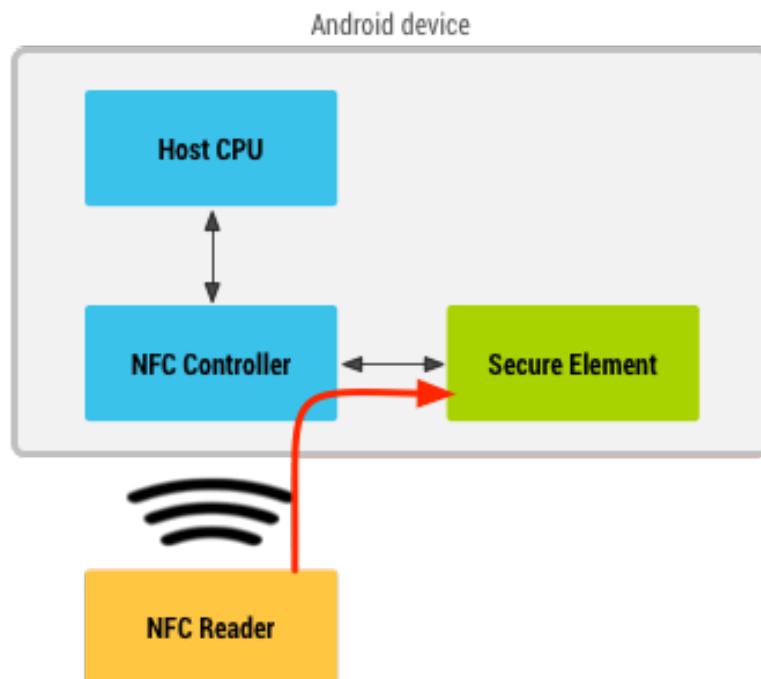


Figura 10: Funcionament de d'un lector NFC amb un mòbil com a HCE

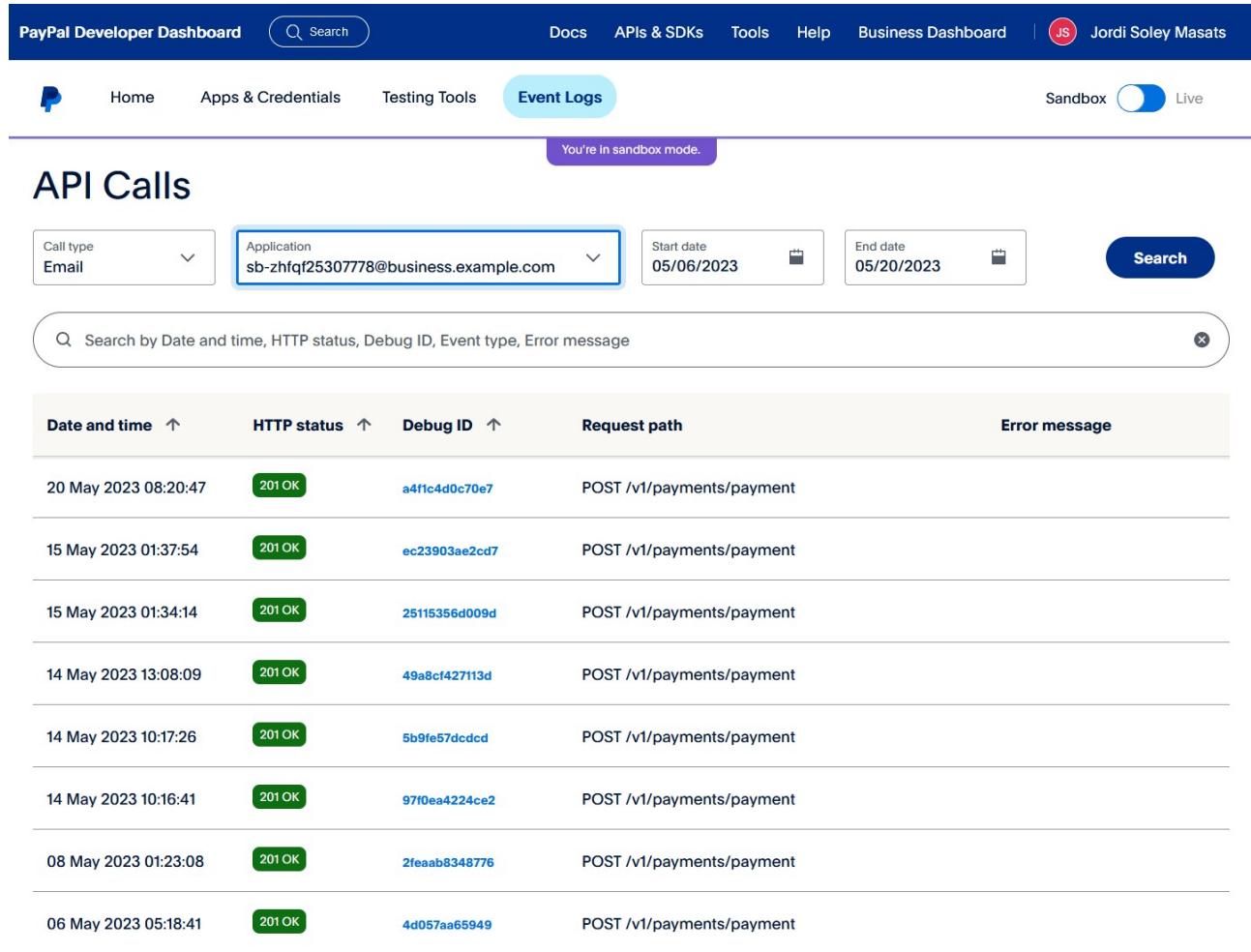
## 6.6 Intregració de PayPal

Com a plataforma de pagament s'ha decidit utilitzar l'API de **Paypal sandbox** que permet pagar amb targeta o amb el saldo d'un compte de PayPal.

En Flutter no hi havia gaires llibreries per a integrar-la, és per això s'ha hagut d'utilitzar una llibreria molt recent amb poques funcionalitats i documentació però que ha satisfet les necessitats requerides: FLUTTER\_PAYPAL V0.0.8 [6]

Permet redirigir a l'usuari a la sandbox de paypal on es genera una pàgina de pagament on mitjançant un codi JSON s'inclouen els articles seleccionats amb els seus preus com també les dades de facturació de l'usuari escrites prèviament per a que no les hagi d'escriure cada vegada.

Així mateix, amb una *clientID* i una *secretKey* del compte developer de Paypal d'un dels membres del grup, s'aconsegueix que les transaccions es realitzin contra aquest compte, per tant els diners seran dispositats en aquest compte quan un client faci un pagament. [20] [19]



The screenshot shows the PayPal Developer Dashboard with the following details:

- Header:** PayPal Developer Dashboard, Search bar, Docs, APIs & SDKs, Tools, Help, Business Dashboard, User: Jordi Soley Masats (JS).
- Navigation:** Home, Apps & Credentials, Testing Tools, Event Logs (selected), Sandbox (Live).
- Message:** You're in sandbox mode.
- Section:** API Calls
- Filters:** Call type (Email), Application (sb-zhfqf25307778@business.example.com), Start date (05/06/2023), End date (05/20/2023), Search button.
- Search Bar:** Search by Date and time, HTTP status, Debug ID, Event type, Error message.
- Table:** Shows a list of API calls with columns: Date and time, HTTP status, Debug ID, Request path, Error message.

Date and time	HTTP status	Debug ID	Request path	Error message
20 May 2023 08:20:47	201 OK	a4f1c4d0c70e7	POST /v1/payments/payment	
15 May 2023 01:37:54	201 OK	ec23903ae2cd7	POST /v1/payments/payment	
15 May 2023 01:34:14	201 OK	25115356d009d	POST /v1/payments/payment	
14 May 2023 13:08:09	201 OK	49a8cf427113d	POST /v1/payments/payment	
14 May 2023 10:17:26	201 OK	5b9fe57dcdd	POST /v1/payments/payment	
14 May 2023 10:16:41	201 OK	97f0ea4224ce2	POST /v1/payments/payment	
08 May 2023 01:23:08	201 OK	2feaab8348776	POST /v1/payments/payment	
06 May 2023 05:18:41	201 OK	4d057aa65949	POST /v1/payments/payment	

Figura 11: Panell de crides a la API de PayPal

## 7 Docker

Docker és la integració més important del nostre projecte, ja que a partir d'ella pivota tot el Sistema d'integració i desplegament continu que expliquem en els següents apartats. Per entendre com s'organitza aquest Sistema automatitzat hem de ser conscients de l'arquitectura actual.

Com hem explicat en els anteriors apartats, disposem de dos Dockers que encapsulen la API i la Base de Dades. La segona es manté estàtica en el Servidor sense redesplegar-se ja que ens interessa conservar les dades entre diferents releases de l'API. Tot i això, l'API no es manté constant i el Docker s'ha d'anar regenerant amb certa freqüència i per ajudar-nos en aquesta tasca farem servir Dockerhub.

L'encapsulament de l'aplicació en contenidors ens proporciona una sèrie de beneficis que poden millorar significativament la gestió i la distribució del nostre Sistema. En primer lloc, Docker permet la creació de contenidors lleugers i portables, que poden ser implementats fàcilment en una gran varietat de plataformes i entorns de producció. Això significa que l'aplicació ens serà més fàcils de distribuir i migrar(si algun dia ens interessi).

Un altre avantatge important de la Dockerització és l'escalabilitat i eficiència. En comparació amb altres solicions de virtualització, Docker ofereix un temps de desplegament molt ràpid i un menor consum de recursos, el que significa que els contenidors Docker poden ser escalats fàcilment(com veurem en els següents apartats).

L'únic problema que vam experimentar en aquesta fase va ser com iniciar la Base de Dades amb els Documents(equivalents a les taules d'una base SQL) requerits quan aturavem el contenidor i el tornàvem a posar en funcionament.

Per últim, un dels aspectes més importants de Docker és la **seguretat**, ja que aïllem els contenidors de les altres peces del Sistema, deixant només les connexions estrictament necessàries (entre API i BD).

## 8 Docker Compose

Docker-Compose és una eina molt útil per a organitzar contenidors Docker i enllaçar-los fàcilment. Vam fer-la servir en una fase inicial del projecte. Tot i no utilitzar-la actualment seria una solució perfectament bona tot i que no seria tant escalable com l'actual.

Ens va permetre enllaçar els dos Dockers principals que em explicat en l'apartat anterior gràcies al fitxer de configuració docker-compose.yml on vam poder detallar com volíem que fos la nostra xarxa Docker explícitament, amb tasques com dependències en el desplegament, IP's dels contenidors i ports oberts.

Docker-Compose ens permet connectar grups de contenidors i gràcies a això vam poder crear una xarxa Docker entre els contenidors i fer que es connectessin entre ells.

## 9 Jenkins

Jenkins és l'eina que em seleccionat com a motor de la nostra Integració Continua, normalment conegut com a CI (Continuous Integration). En el principi del projecte vam prendre decisions com seleccionar aquesta sobre altres competidors com podria ser Gitlab CI pels motius que expliquem a continuació.

En un principi no disposàvem de CD (Continuous Deployment) i el desplegament es feia mitjançant l'eina del punt 5, i un Makefile que vam redactar de suport. Per fer això havíem d'accendir d'alguna manera a la shell i tornar a desplegar els contenidors Docker. Degut a l'infraestructura de les màquines on realitzem els desplegaments ([nattech.fib.upc.edu](http://nattech.fib.upc.edu)) necessitavem una shell per poder fer el desplegament correctament i aquest accés ssh està sota l'accés VPN, que no podem conseguit de forma automatitzada pel requisit actual del two-factor authentication.

La solució pràctica a aquesta problemàtica va ser col·locar un Jenkins al servidor (dintre de la VPN) [9] escoltant gràcies a un webhook al repo[11] i cada cop que rebés una notificació executar les comandes[10] (git pull i make restart) que clonaven l'última versió del repo i el redesplegaven en un Docker com s'ha comentat anteriorment.

A mesura que el projecte ha anat avançant, s'ha anat adaptant la funcionalitat del Jenkins per verificar la seva interconnectivitat amb altres components que han esdevingut indispensables per al Sistema[12], que estàn explicats posteriorment en la present memòria.

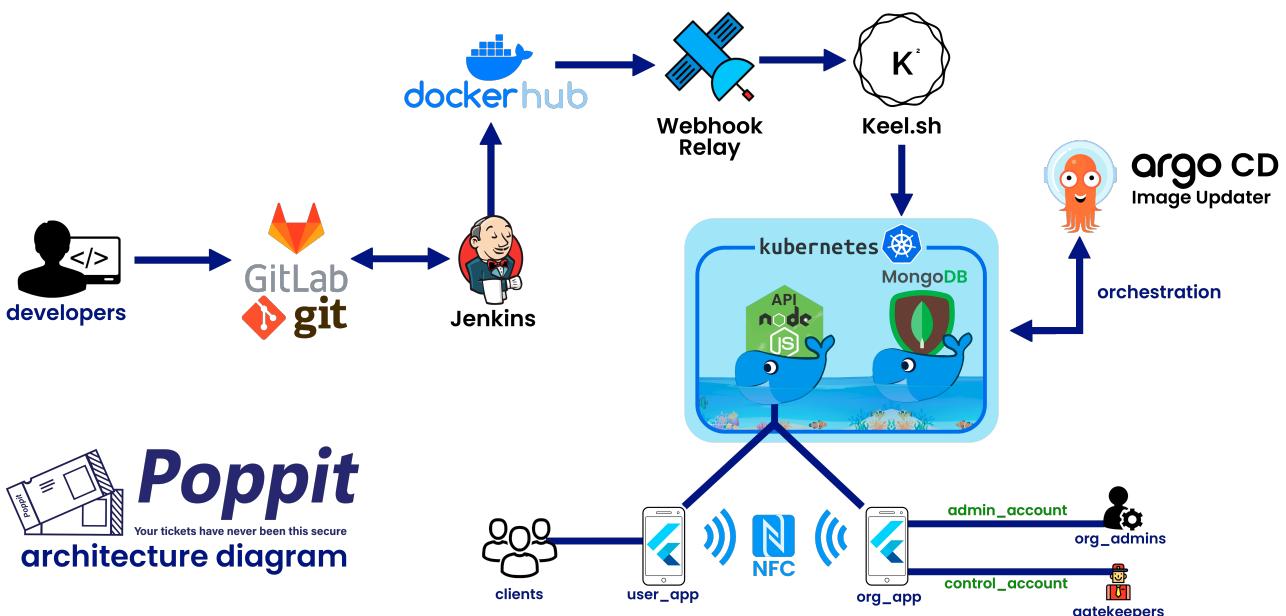


Figura 12: Arquitectura del Sistema

Tal com es pot observar al diagrama de dalt Jenkins és actualment l'encarregat d'agafar els fitxers actualitzats del repositori, construir la imatge Docker i publicar-la al repo DockerHub, per continuar el procés CI/CD. La configuració del Jenkins inclou la redacció d'un Jenkinsfile que indica les diferents etapes que comporten l'evolució de la pipeline, disparada cada cop que es fa un merge a master i es publica una nova Release.

A banda, també s'ha inclòs el plugin Docker i s'han guardat les credencials dels diversos llocs on accedeix la Pipeline. En un futur es podria implementar més funcions com revisions de seguretat al codi o altres eines molt potents que pot integrar Jenkins.

## 10 Clúster de Kubernetes

Amb les eines que ens proporciona la FIB (i sense gastar-nos calés), hem aixecat un clúster de kubernetes als servidors intentant que sigui el més realista possible i ens permeti treballar d'una manera còmoda. Hem usat k3s com a distribució de kubernetes. [14]

Per fer això hem usat algunes eines open source les quals explicarem als següents punts.

En aquest apartat no entrarem en aspectes molt tècnics ni en "tutorials", ja que quedarà massa extens i pensem que no és l'objectiu d'aquesta memòria. De totes maneres deixem alguns enllaços d'interès a la bibliografia.

La següent figura representa el cicle de vida des que es fa un push a docker-hub (amb una nova imatge de la api) fins que es desplega al nostre clúster.

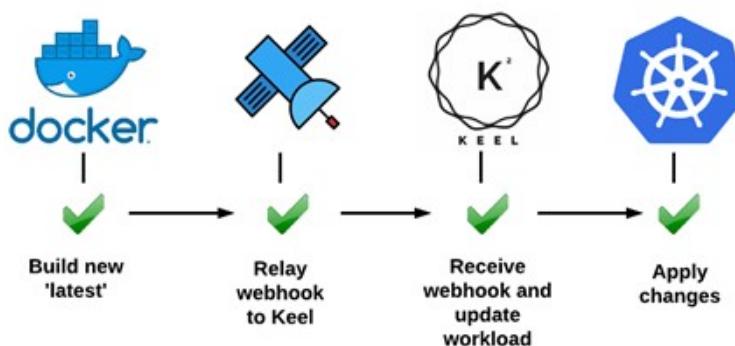


Figura 13: Cicle de vida de la imatge docker

### 10.1 Webhook relay

Webhook relay és l'eina que ens permet comunicar el nostre compte de docker-hub amb keel, la qual explicarem al següent punt. [28]

Aquesta eina ens proporciona dos access tokens, un amb el tag "Key" i un altre amb el tag "secret". Aquests tags són els que utilitzarem per definir el "túnel" del dockerhub fins a l'eina keel al nostre servidor.

```
helm upgrade --install webhookrelay-operator --namespace=push-workflow
  ↳ webhookrelay/webhookrelay-operator \
--set credentials.key=$RELAY_KEY --set credentials.secret=
  ↳ $RELAY_SECRET
```

Listing 1: Comanda per introduir els tags al servidor

## 10.2 Keel

Keel permet al nostre clúster de kubernetes actualitzar les imatges. En el nostre cas, actualitzar la imatge de la api per la nova que s'ha pujat a docker hub i que ens ha arribat al keel gràcies al webhook relay. Utilitzant helm, hem pogut desplegar de manera senzilla aquesta eina al nostre clúster. [15]

Per a que keel funcioni, al manifest de la nostre node-app hem hagut d'escriure unes regles. Keel té moltes combinacions, però nosaltres hem usat aquesta ja que pensem que és la més adient.

```
annotations:
  keel.sh/policy: force
  keel.sh/trigger: poll
  keel.sh/pollSchedule: "@every 3m"
```

Listing 2: Annotations al nostre codi YAML

Aquestes regles迫en un pull de la imatge que tenim a docker-hub cada 3 minuts i la comparen amb la imatge que tenim al nostre servidor deplegada. Keel ens permet fer pull cada minut, però amb el docker-hub estàndard no es permet fer més de 30 pulls l'hora.

D'aquesta manera, des que es puja una imatge a docker hub fins que la tenim operativa al nostre clúster poden passar entre 10 segons (millor dels casos) fins a 3:10 minuts (pitjor dels casos).

## 10.3 Cluster

El nostre clúster està compost de tres serveis. Aquests serveis es declaren a fitxers de configuració YAML i, per tant, es fa un fitxer per cada servei. De la mateixa manera es defineixen els deployments, un per cada app. [3]

→ API service, exposa la api al port 8081 de la màquina, és a dir, serà accessible des de l'exterior mitjançant la drecera <https://poppit.ddns.net:40341/> (explicat al següent punt)

→ MongoDB service, aquest servei no s'exposa. És una Base de dades la qual és accessible només des de dins del servidor. Al deployment de la API hem definit una variable d'entorn per definir a quin contingidor ha d'apuntar la api per poder-se connectar a aquesta base de dades.

→ ArgoCD service, exposa el server d'Argo (explicat més endavant) al port 8080 de la màquina el qual és accessible des de l'exterior a <http://natttech.fib.upc.edu:40340/>

## 10.4 Domini i TLS

Hem creat un domini gratuït anomenat poppit.ddns.net per poder accedir des de fora amb https. Aquest domini l'hem creat gràcies a la eina no-ip. [17] A part de crear el domini, per poder fer servir el TLS, hem creat un certificat gratuït de Let's Encrypt.

Hem definit dos manifests nous al nostre servidor. Un ens serveix per enllaçar el nostre servidor al domini que hem creat i l'altre ens serveix per definir el certificat autofirmat.

D'aquesta manera per accedir al nostre servidor usarem → <https://poppit.ddns.net:40341/>

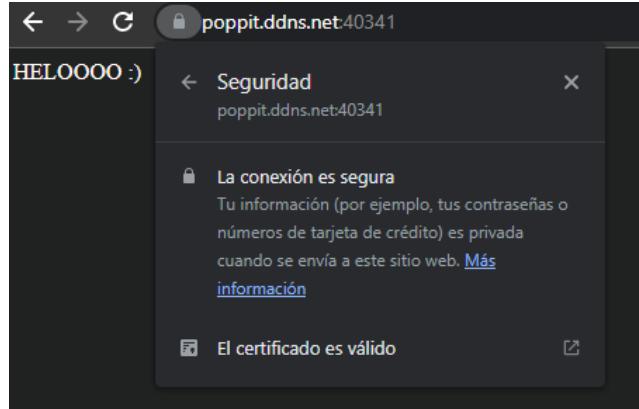


Figura 14: Domini i TLS

## 10.5 Persistència imatges

Per no perdre les imatges (fotografies) cada cop que la imatge de la api s'actualitza i es crea el pod nou, hem hagut de crear dos volums persistents per guardar les fotografies de la api ja que no es poden guardar a la base de dades (un per les imatges de event i un altre per les d'organitzacions). [21]

Això ho hem fet definint volums persistents clàssics de kubernetes (pv i pvc) els quals guarden les fotografies al servidor i es guarden entre sessions.

## 10.6 Orquestració amb ArgoCD

Utilitzem ArgoCD per poder orquestrar el nostre clúster de manera senzilla. [2] ArgoCD ens mostra de forma molt visual el que està passant al nostre clúster en tot moment i d'aquesta manera no haver d'estar executant comandes de consola. Per instal·lar ArgoCD al server hem fet ús de l'eina homebrew. [7]

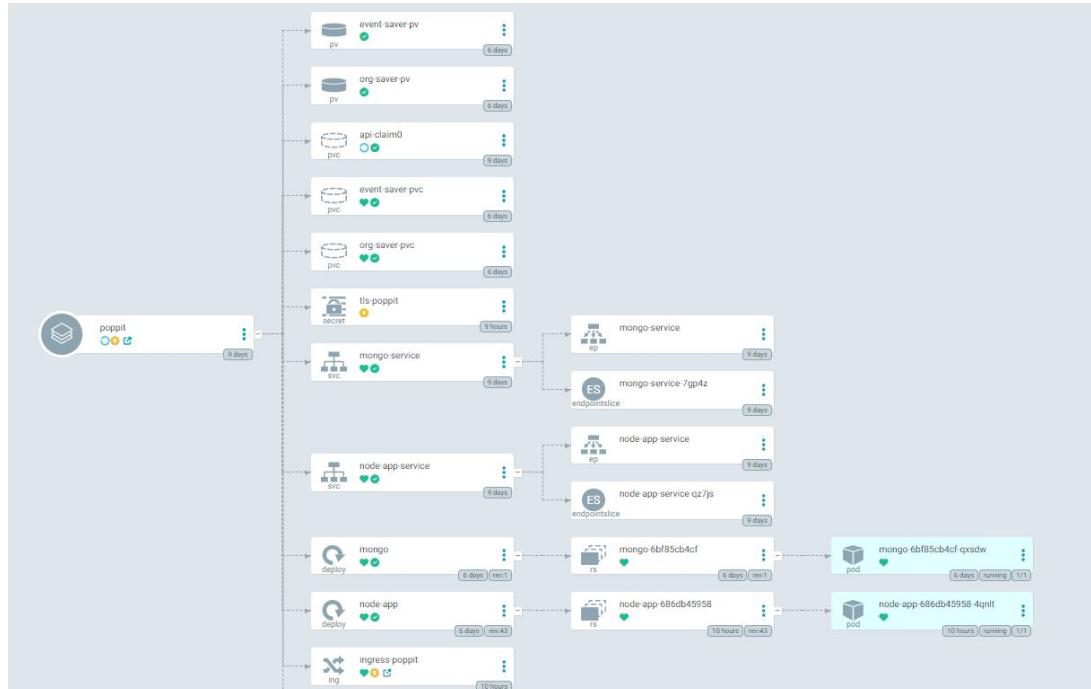


Figura 15: Cluster vist des de ArgoCD

En aquesta imatge podem veure el nostre clúster des de ArgoCD. Veiem els pods actius i els serveis que es poden exposar, en tots dos casos, la api i la base de dades mongoDB. Argocd ens mostra també els logs dels nostres pods per poder veure què s'està fent en tot moment i, en cas d'error, poder detectar-ho de forma fàcil i visual.

## 11 Possibles millores

### 11.1 Afegir mapa events propers

Actualment, la nostra aplicació conté un mapa però aquest no és interactiu ni dinàmic. Aquesta millora es basa en un mapa que mostra els esdeveniments més propers en temps real, és implementar un sistema de filtratge avançat d'esdeveniments. Aquesta millora permetria als usuaris personalitzar les seves preferències i veure només els esdeveniments que més els interessin. El filtratge es basaria en diversos àmbits com per categoria, ubicació, data i hora, popularitat i fins i tot combinades de les esmentades. Amb aquesta millora de filtratge avançat d'esdeveniments, la nostra aplicació proporcionaria als usuaris una experiència més personalitzada i ajustada als seus interessos i necessitats. Permetria als usuaris descobrir més fàcilment els esdeveniments que realment els interessen en temps real, proporcionant-los una millor experiència al descobrir noves oportunitats i activitats que es trobin a prop seu.

### 11.2 Millores NFC

Degut als problemes que hem tingut amb l'integració de la tecnologia NFC a l'aplicació en flutter, no s'ha pogut assolir la comunicació peer to peer directe entre dos dispositius android per a poder validar les entrades dels events. Per això ara mateix utilitzem un intermediari que seria el tag NFC del qual es llegeix i s'escriu. Amb més temps es podria mirar de fer una llibreria nova de flutter que serveixi com a *host card emulator* la qual pugui emular el funcionament d'una targeta, fent que pugui enviar i rebre *APDU commands*, els quals són necessaris per mantenir una connexió amb la targeta. També hi hauria la opció d'aprendre a integrar Kotlin en Dart ja que és possible que d'alguna forma puguin utilitzar-se alhora i fer servir llibreries de kotlin de HCE.

### 11.3 Comprar tickets per amics

Una possible millora per a la nostra aplicació és implementar la funcionalitat de compra de tiquets per als amics. Aquesta característica permetria als usuaris adquirir entrades per a esdeveniments no només per a ells mateixos, sinó també per als seus amics que utilitzen l'aplicació. La seva implementació es basaria en el moment en què un usuari selecciona un esdeveniment i decideix comprar una entrada, l'aplicació oferiria l'opció d'afegir entrades addicionals per als seus amics. Aquesta opció podria ser accessible a través d'una funció "afegir tiquet per contacte". Amb la nova implementació de compra de tiquets per amics, la nostra aplicació proporcionaria als usuaris una manera més amigable i eficient de coordinar la compra i distribució d'entrades. Simplificaria el procés i fomentaria la integració social a través de l'aplicació, permetent als usuaris compartir experiències i gaudir de l'esdeveniment junts.

## 11.4 Implementar ofertes a la revenda

Una possible millora per a la nostra aplicació és implementar la funcionalitat de proposar ofertes a la revenda d'entrades. Aquesta característica permetria als usuaris proporcionar un preu al qual comprarien una entrada d'un esdeveniment i que un venedor de revenda pogues acceptar el preu i es realitzes la revenda. La seva implementació es basaria en un espai on l'usuari que vol comprar el tiquet proporcione un preu, i que tots els venedors de tiquets de revenda d'aquell esdeveniment poguessin veure els preus que proporcionen els usuaris.

## 11.5 Multiplataforma (Android, iPhone)

Una possible millora per a la nostra aplicació seria convertir-la en una aplicació multiplataforma, de manera que no només estigui disponible per a dispositius Android, sinó també per a iOS. Aquesta ampliació permetria arribar a un major nombre d'usuaris i oferir una experiència consistent i accessible en diferents dispositius mòbils.

## 11.6 Dockerhub premium i fiabilitat del servidors

Actualment, dockerhub només ens permet fer pull de la imatge cada 3 minuts. Pagant la subscripció a dockerhub prèmium podríem fer pulls il·limitats de la imatge el que es traduiria en un deploy quasi instantani del nostre servei.

Per altra banda. La fiabilitat dels servidors de la FIB és bastant baixa. Al llarg del desenvolupament de la pràctica hem tingut diferents inconvenients. Se'ns van esborrar les màquines virtuals dos cops i ens vam adonar que els servidors entren en mode "sleep" a la matinada, el que no ens permet treballar de nit. Per això pensem que és important en un sistema real tenir un bon proveïdor de serveis com podria ser AWS, Azure, Google Cloud...

## 11.7 Nom de domini propi i personalitzat

Pel desenvolupament del projecte hem usat un nom de domini estàndard i gratuït que ens ha proporcionat l'eina no\_IP explicada en apartats anteriors. Pensem que de cara a una situació real, seria necessari comprar un domini propi personalitzat.

## 12 Bibliografia

Ordenat per ordre alfabètic.

## Referències

- [1] *Android HCE: Article de HCE en android amb kotlin.* URL: <https://www.zeromolecule.com/blog/host-card-emulation-hce-with-android-and-flutter/>.
- [2] *ArgoCD init: Instal·lació i inització d'ArgoCD a un servidor.* URL: [https://argo-cd.readthedocs.io/en/stable/getting\\_started/](https://argo-cd.readthedocs.io/en/stable/getting_started/).
- [3] *Definition of deployments and services: Documentació per crear serveis i deployments a kubernetes.* URL: <https://sweetcode.io/how-to-create-kubernetes-deployments-and-services-using-yaml-files/>.
- [4] *Diseny login: Informació utlitzada per a crear les pantalles de login.* URL: <https://www.youtube.com/watch?v=Dh-cTQJgM-Q>.
- [5] *Diseny navigation bar: Informació utlitzada per a crear la navitagion bar.* URL: <https://www.youtube.com/watch?v=FEvYl8Mzsxw&t=48s>.
- [6] *flutter-paypal v0.0.8: A simple but powerful Paypal SDK for flutter.* URL: [https://pub.dev/packages/flutter\\_paypal](https://pub.dev/packages/flutter_paypal).
- [7] *HomeBrew Install: Instal·lació de HomeBrew a un servidor.* URL: <https://phoenixnap.com/kb/homebrew-for-linux>.
- [8] *Image Encoding: Solució per fer encode de les imatges.* URL: <https://gist.github.com/hakobera/961513>.
- [9] *Jenkins Installation: Informació de suport per instal·lar i configurar Jenkins a un dels servidors.* URL: <https://www.makeuseof.com/install-jenkins-on-ubuntu/>.
- [10] *Jenkins pipeline trigger: Informació utlitzada per poder fer trigger de la pipeline de Jenkins.* URL: <https://www.youtube.com/watch?v=r5zhTu694Kc>.
- [11] *Jenkins Webhook detection: Informació relacionada amb l'obtenció de dades per part de Jenkins.* URL: <https://santoshk.dev/posts/2022/how-to-setup-a-github-to-jenkins-pipeline-with-webhook/>.
- [12] *Jenkins/ArgoCD connection: Informació utlitzada per connectar Kubernetes amb Jenkins.* URL: <https://www.opsmx.com/blog/how-to-enable-ci-cd-with-argo-cd-and-jenkins/>.
- [13] *JWT Token Security: Documentació per proporcionar seguretat mitjançant tokens JWT.* URL: <https://www.section.io/engineering-education/how-to-build-authentication-api-with-jwt-token-in-nodejs/>.
- [14] *k3s Install: Instal·lació de k3s a un servidor.* URL: <https://computingforgeeks.com/install-kubernetes-on-ubuntu-using-k3s/>.
- [15] *Keel usage: Pàgina web usada com a base per fer servir keel.sh.* URL: <https://keel.sh/docs/#kubernetes>.
- [16] *Key Generation: Resolució de dubtes per generar les claus de l'API i de les Aplicacions amb OpenSSL.* URL: <https://stackoverflow.com/questions/16056135/how-to-use-openssl-to-encrypt-decrypt-files>.
- [17] *no-IP free domain: Domini gratuït per exposar la API amb TLS.* URL: <https://www.noip.com/es-MX>.

- [18] *Official flutter packages*: Pàgina oficial de llibreries de flutter. URL: <https://pub.dev/>.
- [19] *Paypal api*: Paypal developer api documentation. URL: <https://developer.paypal.com/api/rest/>.
- [20] *Paypal sandbox testing guide*: Paypal developer documentation. URL: <https://developer.paypal.com/tools/sandbox/>.
- [21] *Persistent Volume Documentation*: Documentació per crear volums persistents a k3s. URL: <https://kubernetes.io/docs/concepts/storage/persistent-volumes/>.
- [22] *Separation in different files*: Diferenciació d'esquemes a la base de dades. URL: <https://medium.com/@jelaniwoods/mongoose-schemas-in-separate-files-3473de658c84>.
- [23] *SHA256 Encryption System*: Paquet per encriptar i desencriptar peticions amb Node.JS utilitzant claus públiques i privades. URL: <https://www.npmjs.com/package/node-forge>.
- [24] *Tutorial Flutter*: Informació utlitzada per a crear home screen. URL: <https://www.youtube.com/watch?v=GwhpedXmc4M>.
- [25] *Tutorial Flutter*: Informació utlitzada per a crear l'aplicació. URL: [https://www.youtube.com/watch?v=71AsYo2q\\_0Y](https://www.youtube.com/watch?v=71AsYo2q_0Y).
- [26] *Tutorial Flutter*: Informació utlitzada per a instal·lar flutter. URL: <https://www.youtube.com/watch?v=l108UNBwMOA>.
- [27] *Tutorial Flutter*: Informació utlitzada per a introduirnos a flutter. URL: [https://www.youtube.com/playlist?list=PL1\\_hIU4u7P677H9f6zPOHi0z2izkvQq2E](https://www.youtube.com/playlist?list=PL1_hIU4u7P677H9f6zPOHi0z2izkvQq2E).
- [28] *Webhook relay usage*: Pàgina web usada com a base per fer servir webhook relay. URL: <https://webhookrelay.com/>.