Faculdade de Design, Tecnologia e Comunicação
Universidade Europeia

pacaward

Project Software Engineering Report

Francisco Cordeiro 50037301

June 2020

Table of Contents

1. Software Project Description

1.1. Project overview

This project was proposed by Fidel API and its aim is to build a simple card-linked offers Android app for the people who shop often with a credit/debit card. The app allows the users to view the available offers and their locations; link multiple cards; list the cards already linked; list the card-linked transactions and show a push notification on a new transaction. Integrates Fidel tools (API and SDK) and makes use of many Amazon Web Services like Cognito, Lambda and Simple Notification Service.

1.2. Purpose of the project

1.2.1. Business domain

In a card-linked offers app it is possible to receive instant cashback when buying a product that has a discount in the app, just by linking a credit/debit card.

1.2.2. Main project goals

The main goal of this project was to create a card-linked offers android mobile app that is easy to use and transparent. The app is very simple, containing three main screens: offers, transactions and cards. This simplicity makes it easy for the users to navigate around without needing any aid or user manuals.

1.3. Project background and scope

1.3.1. Context of the work

Consumers often have the problem of receiving discount offers for things they do not really need. Moreover, retailers also have problems when it comes to targeting the right customers for their discounts.

Card-linked offers (CLO) are Online-to-Offline (O2O) systems, meaning the online marketing leads to an offline (physical) experience, helping mitigate the problems for both entities. Firstly by getting consumers behavior data (with their consent), analysing it, enabling the retailers to make specific marketing campaigns. Secondly by having personalized offers to each consumer, applying the discounts automatically on each purchase just by using the credit or debit card.

Customer preference is driven by convenience, therefore, connecting offers to payment cards makes the experience frictionless - there are no additional steps at checkout and no codes, coupons, vouchers or membership cards. This reduces the risk of customer drop-off and disengagement. Offers can also be tailored and more responsive to preference, as opposed to the other older physical methods which are static, take a blanket approach and are therefore often irrelevant.

Pacaward is a CLO Android application that integrates Fidel API. This is a card-linked API that lets developers create web and mobile applications for linking bank cards with reward services. Fidel enables the app to connect credit and debit cards and monitor transactions. This way, when a user makes a purchase at a participating store with a linked card, the API sends the transaction information to the application's server in real time through webhooks.

Fidel runs on top of global payment networks, so it does not require changes to existing merchant infrastructures. All the Payment Card Industry (PCI) compliance requirements are managed by Fidel, removing the weight of dealing with sensitive user data from the application.

1.3.2. Current situation

The world is currently facing a global pandemic which is leading to a global economic crisis. This made an impact in social behaviours and spending habits caused by reduced consumer confidence.

With the restrictions and social distancing consumers are embracing online shopping, reconsidering consumption habits and reallocating budgets. Many merchants had to shut down entirely during the lockdown, others had to manage around the strict rules.

The fallout from this situation will have disastrous knock-on effects to the livelihoods of millions of people across the globe, as retail employment remains at risk. That, in turn, will trigger a rise in saving - a common response to a crisis. However, that habitual, excessive hoarding of cash will mean that economic activity remains depressed for longer, exacerbating the problem. Counter-intuitively, less spending now will result in less disposable income in the future. Beyond the direct economic risk from the virus, a lack of demand that extends beyond lockdown may be the most serious risk to the countries' outlook.

This created an unprecedented pressure on consumers and merchants alike. A solution that would benefit both is card-linked offers. A CLO app, makes users torn to pay their purchases with a cashless method (bank cards) instead of the physical currencies. The last method is problematic since the coronavirus easily spreads through bills and coins.

Recent findings have indicated that cashback payments increase the likelihood of consumers returning to the application for additional purchases, and also increase the size of the purchases themselves.

Card-linking is the best way to deliver a rewards programme since vouchers are simply not nuanced enough to complement individual customer habits and choices. By contrast, card-linking enables targeted distribution of rewards to strategically stimulate spending.

1.3.3. Competing products and benchmarking

Card-linked offers is a booming market as a result of the shift to cashless (growth of 20% per year), thus there are many apps related to this, the most relevant being:

- Acorns (https://www.acorns.com/) - in every purchase with a liked-card, rounds up to next dollar, giving the user a choice to save or invest their change.
- Dosh (https://www.dosh.cash/) - partner of more than 1000 stores and restaurants, has a referral system where users can earn money when a friend links a card.
- Drop (https://www.earnwithdrop.com/) - works with points instead of direct cashback.

Although there are many CLO applications like the ones presented before, all applications differ in their partners, meaning there will be distinct retailers, therefore different offers. Apps also differ in the way they offer their discounts, some work with direct cashback, others use a points system forcing the users to spend it on other offers.

1.4. Mockups and prototypes (early prototypes)

The mockups drawn at the start of the project are exposed in Figure 1. These were designed based on the project owner ideas and approved by the same.

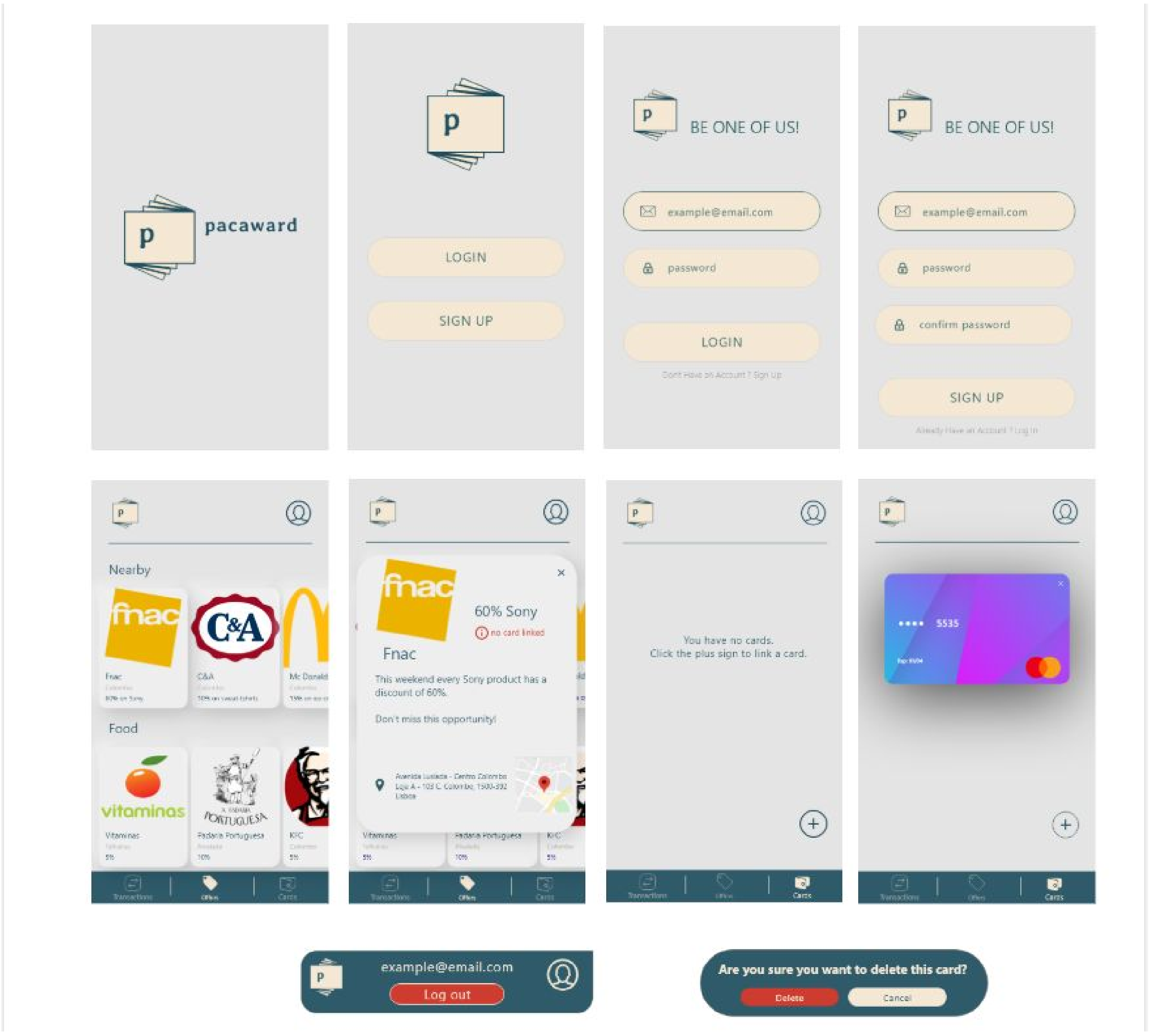


Figure 1 - Early Mockups

1.4.1. Product scenarios

Main scenario

After logging in, the user sees a list of offers available, links a new card, sees a list of his linked cards and the transactions he has made.

1. User logs in;
2. When entering the app, the user has a list of all offers available;
3. By clicking on Cards the user is redirected to cards screen;
4. User sees a lists of linked cards (if no cards are linked a message will be displayed);
5. By clicking in the plus sign to add a new card and is redirected to Fidel SDK;
6. Fills the information required and submits;
7. Is redirected to the Cards screen where the new card is already visible;
8. User clicks Transactions;
9. It is displayed a list of the transactions made by the user filtered by date.

Secondary scenarios

1st.

User makes a transaction with a linked card and receives a push notification.

1. User makes a transaction with a linked card;
2. User is logged in on a device;
3. Fidel webhook is triggered;
4. Push notification is sent to the user's device.

2nd.

User is on the home screen, clicks on an offer and it is expanded, showing detailed information and a location preview.

1. User is in offers screen;
2. User selects one of the offers on the list;
3. New screen opens with the offer details.

1.5. Stakeholders

1.5.1. Development team

Francisco Cordeiro

1.5.2. User

People who make purchases with credit/debit cards.

1.5.3. Project Owner

Fidel API

1.5.4. Project Manager

Prof. Jacinto Estima

Prof. José Vasconcelos

1.6. General constraints

1.6.1. Solution constraints

There were some solution constraints requested by the project owner. These had the purpose of expanding the knowledge of the project developer, preparing the latter to the job market. These constraints are:

- The app should be written in Java;
- Use Fidel API test environment and playground;
- Integrate Fidel native SDK;
- Use AWS or Google Cloud or Microsoft Azure to support the real-time webhook functionality;
- Git should be used for version control.

1.6.2. Schedule constraints

The project must be accomplished in a 18 week period.

1.7. Conventions and definitions

1.7.1. Key terms

CLO - Card Linked Offers

1.8. Relevant facts and assumptions

There are two relevant facts about the integration of Fidel API:

- Since all information displayed to the user is always fetched from Fidel API, if the API status is down, the app will not show any information. For this to happen all objects would have to be stored in the database, increasing the cost and threatening the scalability of the application. One alternative would be to pay for a better Service Level Agreement (SLA) with Fidel API where they would make assurances on the availability of their product.
- The international expansion of this app is limited to the supported countries by Fidel API.

2. Software Requirements

2.1. Actors and use cases

2.1.1. Actors

The application has only one actor named **User** that makes reference to every user.

2.1.2. Use case list

As depicted in Table 1, there are five use cases. These are:

- Use Case 0 : User sees offers by category - in the offers screen there is a list of all available offers divided by category;
- Use Case 1: User sees his transactions - in the transactions screen there is a list of all transactions in which the user received cashback ordered by date.
- Use Case 2: User sees his cards - in the cards screen the user has a list of all the cards that were linked. For safety reasons only the last four digits of the card number are shown.
- Use Case 3: User adds new card - there is a button in the cards screen that takes the user to the Fidel SDK UI where this can link his card safely.
- Use Case 4: User receives a push notification on a new transaction - Whenever the user makes a transaction that has cashback, and is logged in on a device a push notification is sent.

UC	Name
0	User sees offers by category
1	User sees his transactions
2	User sees his cards
3	User adds new card
4	User receives push notification on new transaction

Table 1 - Use Cases

2.1.3. Use case diagrams

The use case diagram (Figure 2) models the functionalities (functional requirements) of the actor in a high level approach.

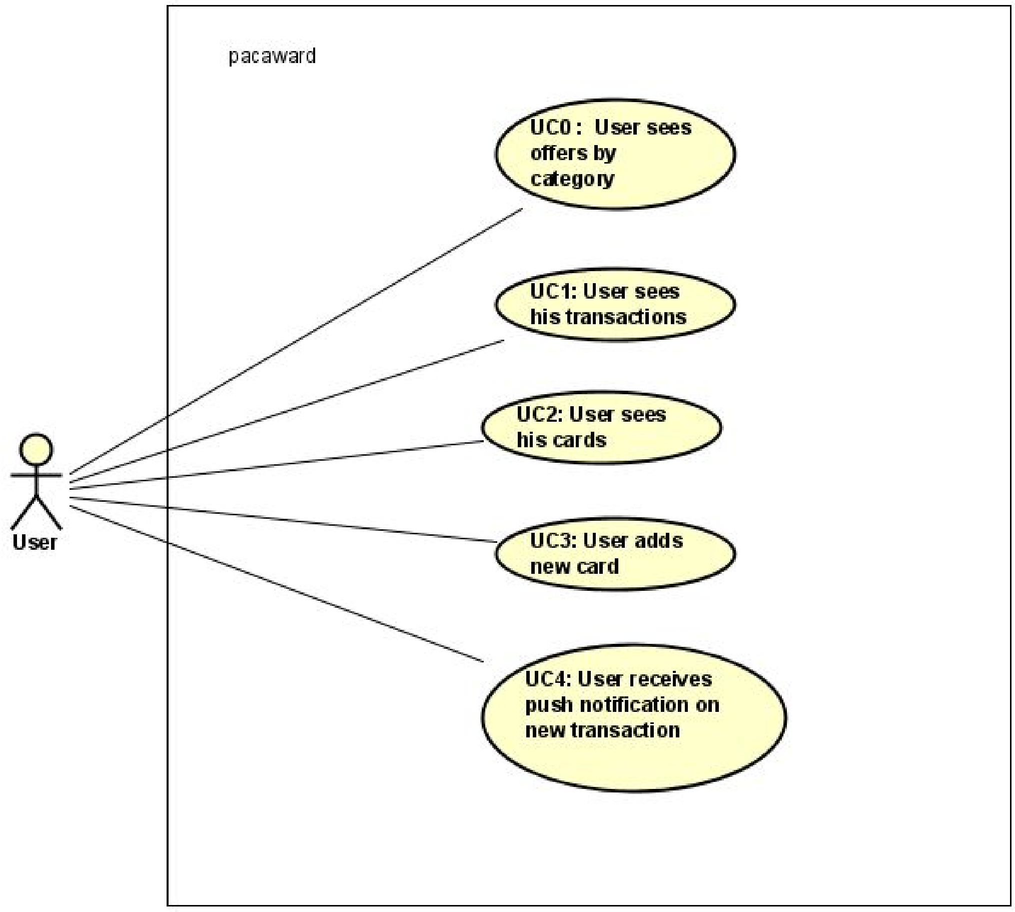
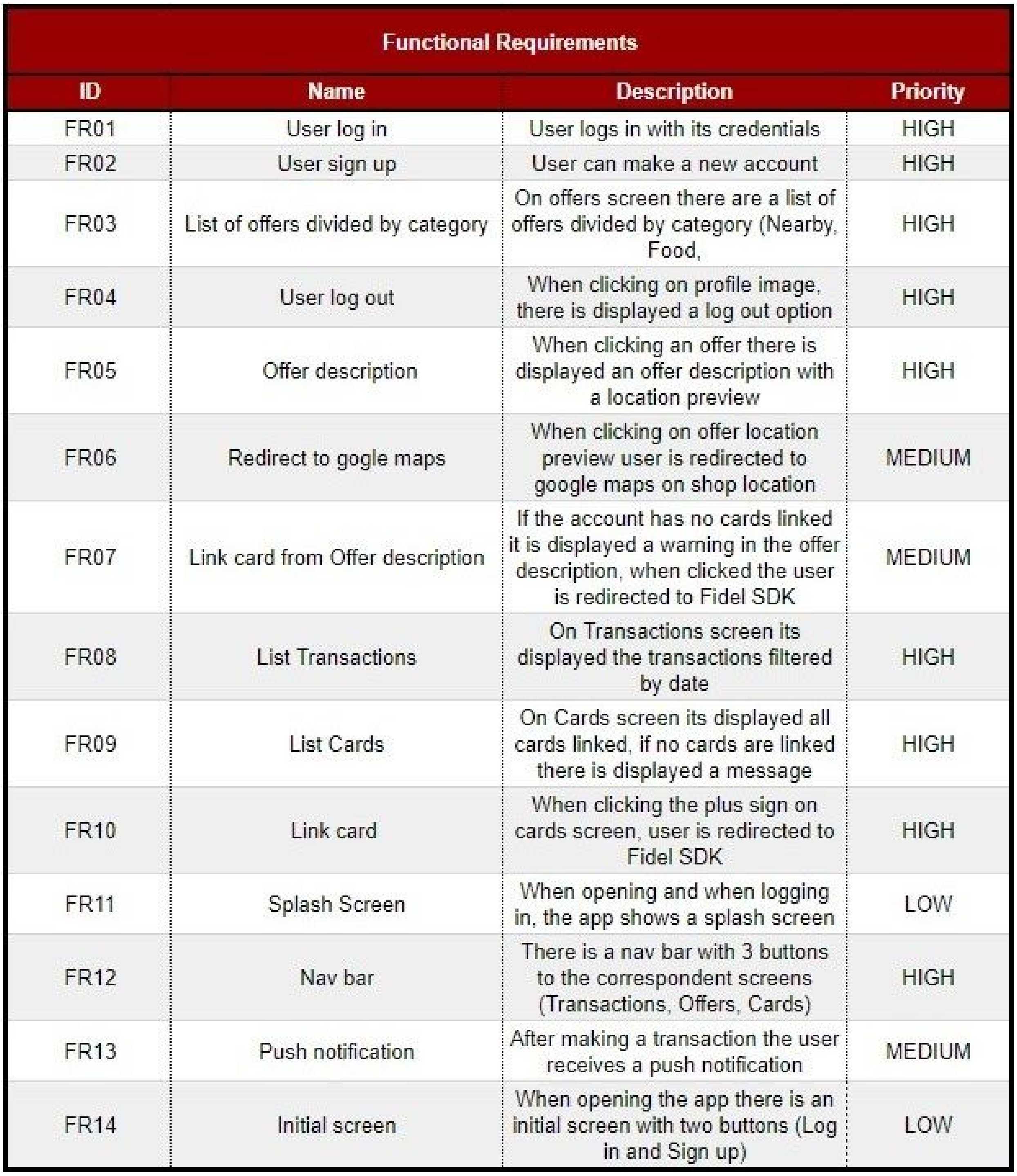

pacaward
UC0 : User sees offers by category
UC1: User sees his transactions
UC2: User sees his cards
UC3: User adds new card
UC4: User receives push notification on new transaction
User

Figure 2 - Use Case Diagram

2.2. Domain and functional requirements

Table 2 shows the list of functional requirements and their priorities. High priority requisites are a *must-have,* Medium priority a *nice to have* and Low priority a *wish to have.*



Functional Requirements			
ID	Name	Description	Priority
FR01	User log in	User logs in with its credentials	HIGH
FR02	User sign up	User can make a new account	HIGH
FR03	List of offers divided by category	On offers screen there are a list of offers divided by category (Nearby, Food,	HIGH
FR04	User log out	When clicking on profile image, there is displayed a log out option	HIGH
FR05	Offer description	When clicking an offer there is displayed an offer description with a location preview	HIGH
FR06	Redirect to gogle maps	When clicking on offer location preview user is redirected to google maps on shop location	MEDIUM
FR07	Link card from Offer description	If the account has no cards linked it is displayed a warning in the offer description, when clicked the user is redirected to Fidel SDK	MEDIUM
FR08	List Transactions	On Transactions screen its displayed the transactions filtered by date	HIGH
FR09	List Cards	On Cards screen its displayed all cards linked, if no cards are linked there is displayed a message	HIGH
FR10	Link card	When clicking the plus sign on cards screen, user is redirected to Fidel SDK	HIGH
FR11	Splash Screen	When opening and when logging in, the app shows a splash screen	LOW
FR12	Nav bar	There is a nav bar with 3 buttons to the correspondent screens (Transactions, Offers, Cards)	HIGH
FR13	Push notification	After making a transaction the user receives a push notification	MEDIUM
FR14	Initial screen	When opening the app there is an initial screen with two buttons (Log in and Sign up)	LOW

Table 2 - Functional Requirements

2.3. Non-functional requirements

2.3.1. Performance requirements

The response time of the transitions between screens should be less than two seconds.

2.3.2. Security requirements

The app should have middleware to connect to the database in order to make the connections secure and assure the database remains secret.

2.3.3. Scalability and extensibility requirements

The app should have middleware to connect to the database since having *n* users directly connected consumes more resources.

2.3.4. Privacy requirements

To assure user data is private AWS Cognito User pools service is used to store sensitive data (email and password), letting the app work only with user references (user id).

2.3.5.Usability requirements

Three main colors are used throughout the app to assure consistency. Buttons are responsive, the current screen is highlighted in the navigation bar and important actions have a red background (e.g. deleting a card) to get special attention from the user.

2.3.6. Internationalization requirements

The app supports multiple currencies and can easily be translated to other languages.

3. Software Design

3.1. Proposed software architecture

3.1.1. General architecture

Figure 3 represents the general architecture of the software divided in:

- Presentation Layer - user interfaces;
- Business Layer - receives information from the Presentation Layer and sends it to the middleware. Moreover, gets the data received from the middleware and makes an HTTP request to the Fidel API, processes the information and sends it to the Presentation layer;
- Middleware - the connection between Business Layer and Data layer;
- Data Layer - where the information is stored.

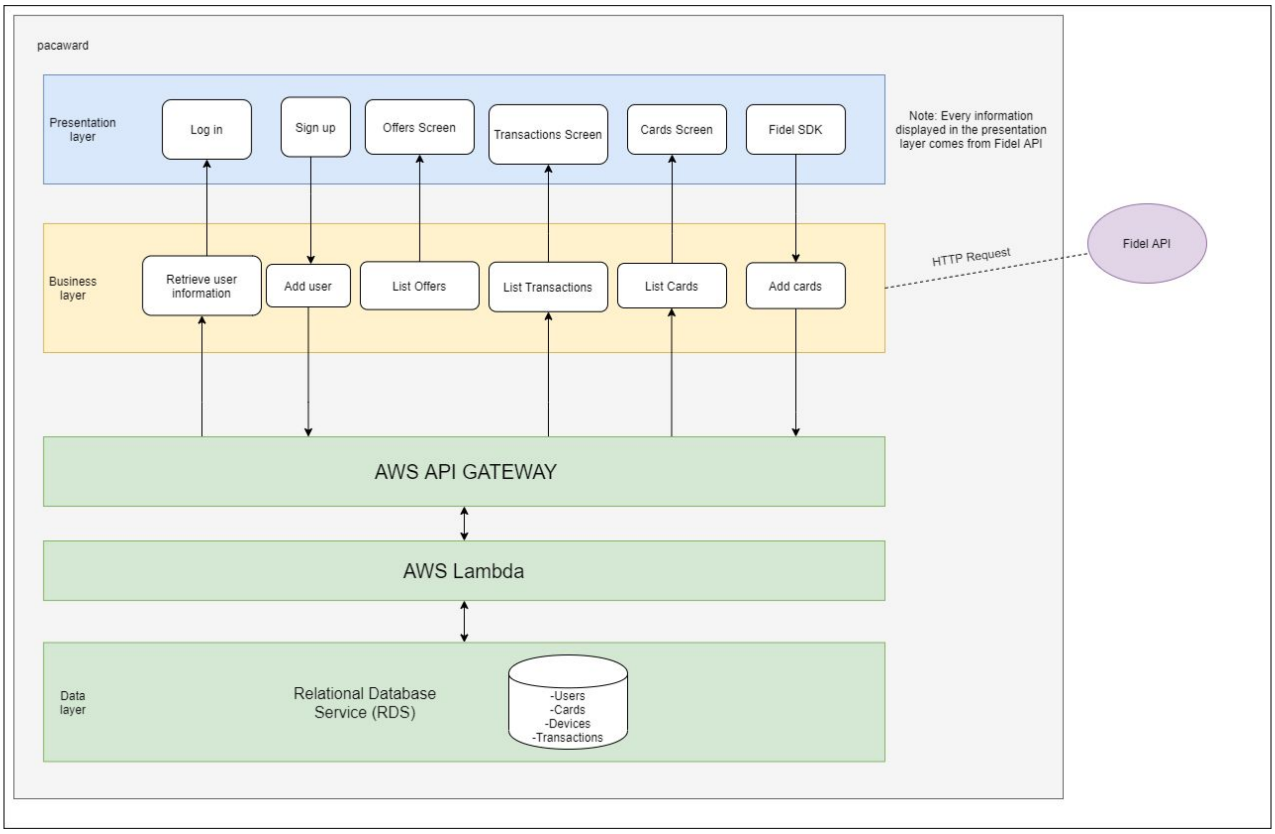


Figure 3 - Project Architecture

3.1.2. Subsystem decomposition

- AWS System

As described in Figure 4, the only communications with the application are with the Amazon Cognito - that manages the authentication, and the Amazon API Gateway - that integrates the REST API.

The communications within the AWS services start in the API Gateway communicating with the AWS Lambda functions. These then connect to the Amazon RDS database and read/write the data.

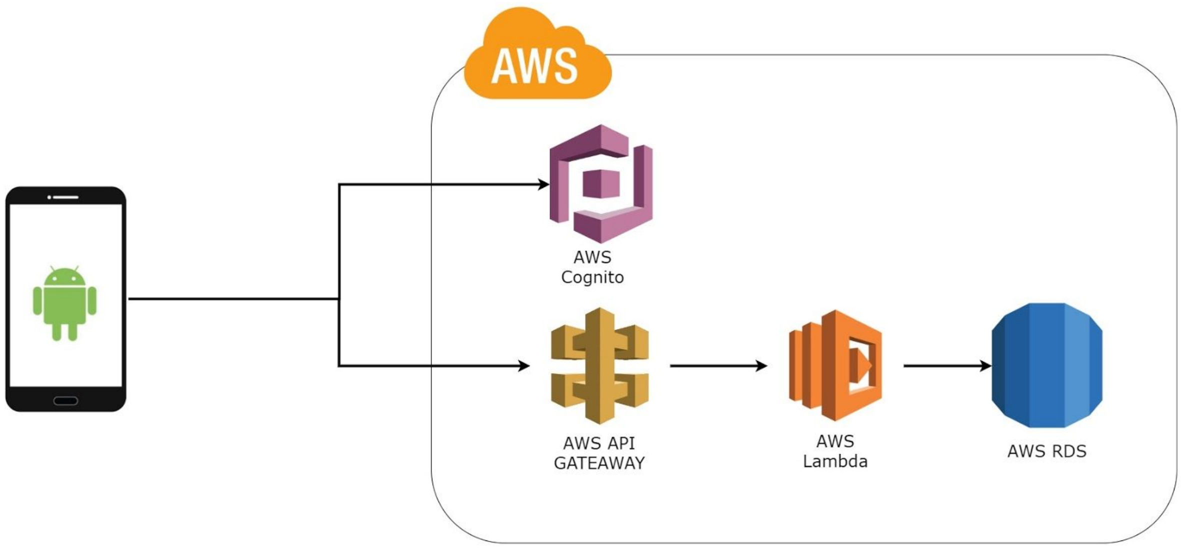


Figure 4 - AWS System Architecture

- Notifications Webhook

The architecture described in Figure 5 represents the creation process of the push notification. Starting when the webhook is triggered by a new transaction, sending an object with the transaction information to an AWS Lambda through AWS API Gateway. In the Lambda function the information is processed and a message is created with AWS SNS, that reaches the device with the support of Google's Firebase Cloud Messaging.

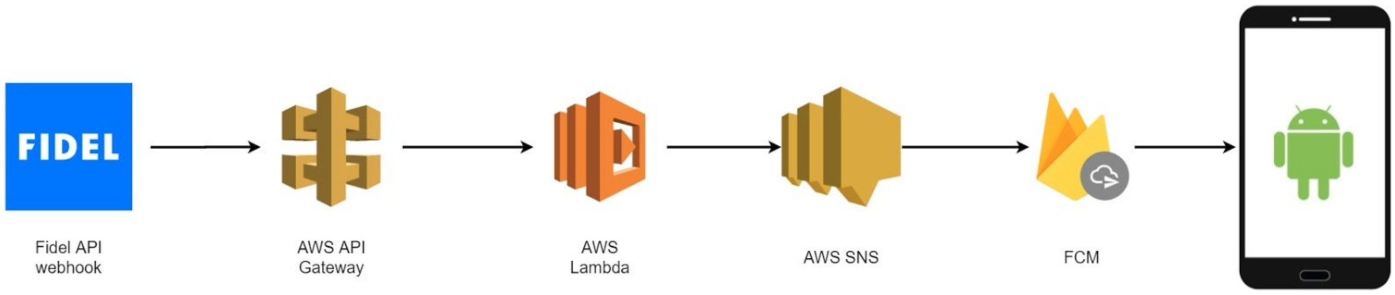


Figure 5 - Notifications Webhook Architecture

3.2. Domain Model

The domain model is represented in figure 6. This model allows the developer to have an high level view of the objects and helps construct the database schema.

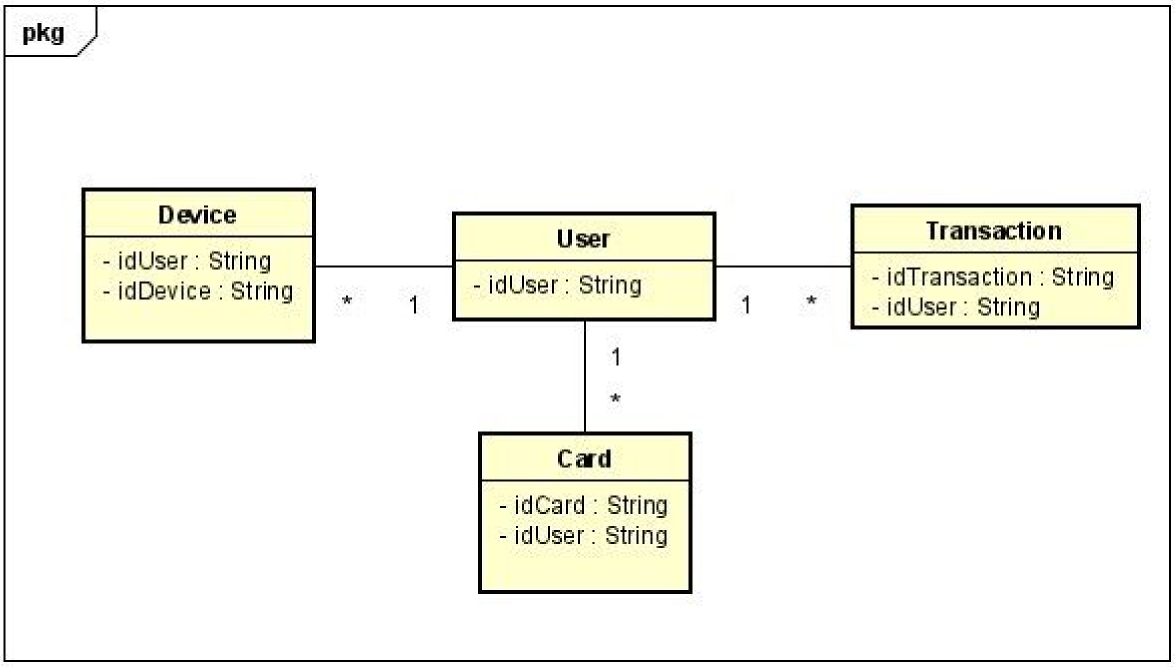


Figure 6 - Domain Model

3.3. Interface flow

Figure 7 represents the interface flow, requested by the project owner to help the developer keep track of the screen transitions when constructing the application.

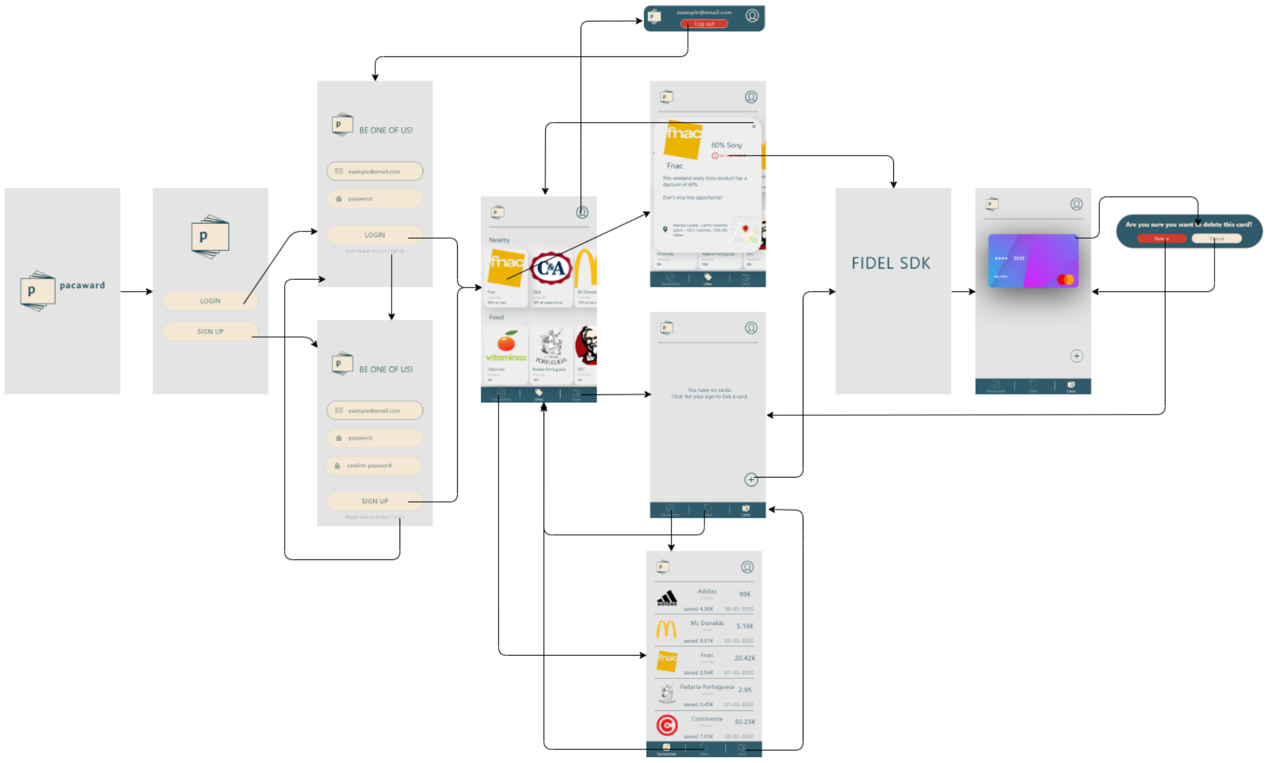


Figure 7 - Interface Flow

4. Software Construction

4.1. Integrated development environment

Project developed with the official IDE for android applications development, Android Studio. The IDE makes it easy to install libraries through the gradle file, open source, has Android emulators and enables debugging as the application runs. One of the big issues is the memory consumption of the IDE.

4.2. Middleware programming

AWS lambdas support serverless functions, which scales automatically, meaning it processes always the amount of data it is given, no matter the size. This allows a reduction of complexity and costs, since only the consumed computation time is paid.

In this project, these functions are developed in node.js and serve as a means of communication between the REST API and the Database.

The push notifications were created with a webhook supported by Fidel API, which sends an object to a lambda function with the information of the transaction. This lambda then communicates with AWS Simple Notification Service which sends a message to the android device of the user that has the card connected to the app through Google’s Firebase Cloud Messaging.

4.3. Unit testing

Every functionality was tested after being implemented. This is important to assure everything works as intended in each step of the development and therefore it is easier to detect faults and correct them.

5. References

[1] Vana, P., Lambrecht, A., & Bertini, M. (2018). Cashback Is Cash Forward: Delaying a Discount to Entice Future Spending. *Journal Of Marketing Research*, *55*(6), 852-868. doi: 10.1177/0022243718811853