Lluís Hernández, Óscar Subirats, Judit Isern, Sergi Olives

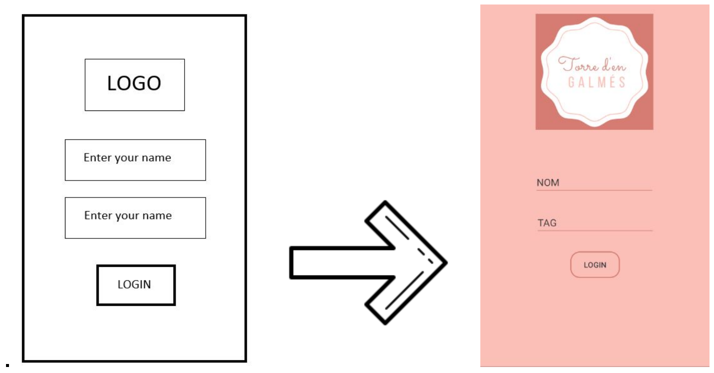
Gamification of a Prehistoric Village - Technical Report

For our project of the gamification of a prehistoric village, inspired in Torre d'en Galmés (Menorca) we developed a mobile game in Android Studio with the main goal of creating an interactive experience to improve the visit of a cultural location and learn while playing. Our demo consists on a small game that is part of the full experience. The game consists in shooting a tennis ball from a sling (like the prehistoric warriors in Menorca) to a wooden cow that is the target. When the player hits the target three times (the ball contains a tag that is detected by an antenna at the target) it wins the game and finishes the demo. The player has an assigned lane and two antennas are reserved exclusively for that player, one at the target, the other at the player location. The ball can be "active" or "inactive" in our database so when it reaches the target, the player has one point but the ball become inactive, does not add any more points and needs to comeback to the player location to be active again, be thrown and score one more point.

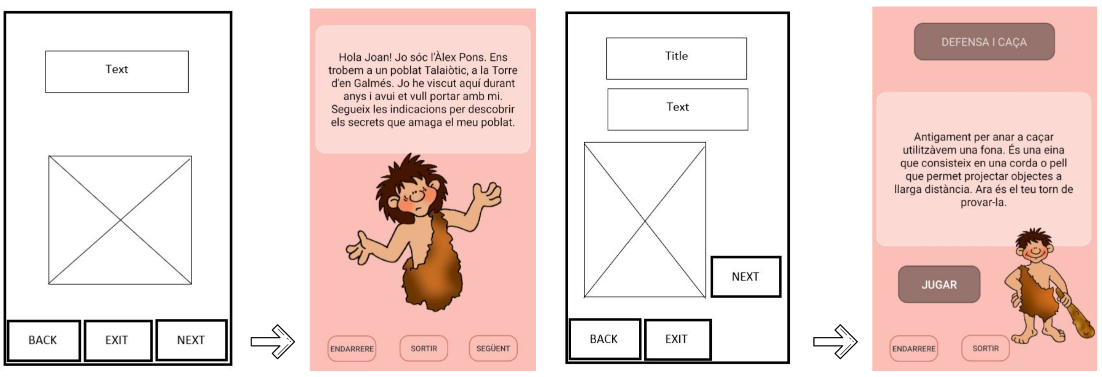
UI Design

In the creation of the user interface we have taken into account from the first moment that it had to be visually very pleasant, since our application is designed for children. We chose to make the application with the pink background because it is a color that is associated to the youth, as our application. On the other hand, it is a color that is related to kindness, delicacy and softness, concepts that are very suitable for dealing with children. The color of the letter is mostly black, and thus it has more contrast with the rose and is easier to read. Except for the titles of the games, which are white with a black background (also facilitating the user's reading)
We created a character, Alex, to make the game closer to the child. The name of the character we chose was because it is a non-sexist name that can be related to both boys and girls, then the user can feel more comfortable. He is a character who speaks directly with the user and thus he delves more into the game.
To make the screens we divided them into 3 general types:

1. Initial screen: where the user registers. On the screen we find where the user enters their name and their TAG, and a button to register.



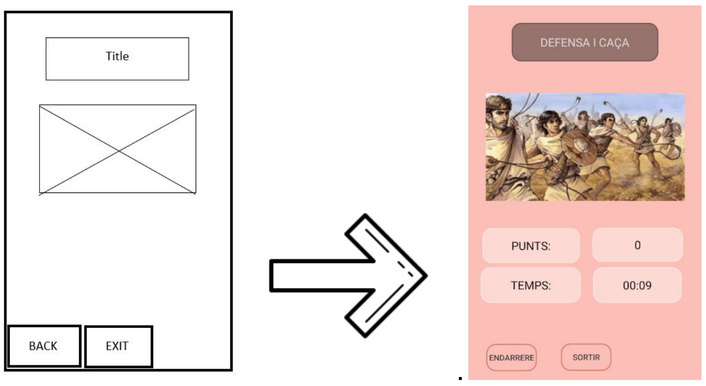
2. Information screens: where the user receives information for the character, either by how to play or where to go. In these screens we always find the character, a text and 3 buttons (back, exit, next). In the case of information about the games, the "next button" is the one that that is written "Jugar", but it does the same function.



Hola Joan! Jo sóc l'Àlex Pons. Ens trobem a un poblat Talaiòtic, a la Torre d'en Galmés. Jo he viscut aquí durant anys i avui et vull portar amb mi. Segueix les indicacions per descobrir els secrets que amaga el meu poblat.

Antigament per anar a caçar utilitzàvem una fona. És una eina que consisteix en una corda o pell que permet projectar objectes a llarga distància. Ara és el teu torn de provar-la.

3. Game screens: the screens where the user is playing, in each game they are different. In the example of the demo we find that there is a timer and an accountant, but it would not be like this for all games



Database design
The database for the demo is a part of a much bigger database of the whole planned application, but we reduced it for the sake of the demo. It contains 3 different classes: users, games and items. The idea here is that when a user enters his/her name and an id that receives at the start of the experience (with these, we can track a given user at any moment) it creates a new instance in the database (for the demo, just a single user will be present in the database). At this moment, we also instantiate a game instance and 10 item instances (one for each ball that is going to be thrown in the game) and each item instance will be linked to an array contained in the game instance. The user will have values for his/her name, the given ID (primary key) and the game (foreign key) he or she is currently at (it can be null if transitioning between games), also he or she will have a lane assigned because all games have different lanes for multiple people to play together and a score that will represent the current score for the player and if he or she is in a game and reaches the win condition of the game, the player will win. For the game it will have an id as a primary key, name, description, location as strings, the win condition or score needed to win the game and proceed with the experience and finally the container for the objects needed for the game (foreign key). And finally, for the items they will have a primary key id, the type and if that item is active or not, this way we can make items do different things depending on which antenna reads them.

XML solution
To parse xml, we have chosen the XMLPullParser (an interface that defines parsing functionality) as it works very well with android studio, being able to parse an xml and search the information in it. First we tried to parse an xml in local, which was more or less easy to do. Then we had to do it parsing from the web and inside our android studio application, which was more difficult as we had to connect to an specific url and use the AsyncTask of android. The AsyncTask (as the name says) creates another not synchronized thread in the application that does an specific task and then it returns a result. We managed to create a RetrieveFeed class, that extends AsyncTask, and call there the XMLPullParser, capturing what we needed (the epc of the tag and the port where it was read). The

xml parser acts like a depth tree, extracting the information that we want looking in the name of the tags. Finally we grab all the information in a list of items and return it to the main thread.

We create a new RetrieveFeed every 5 seconds, and check the status of the AsynkTask every time we enter in the function to see if we can grab the information in items (if the task has finished) or we can't (the task has not finished and the list of items has not been filled in).

Solution for connecting and to query the RFID reader

To query the reader, first is necessary to assemble the environment. We have to connect the 4 antennas to the reader, then connect via ethernet the reader with the transformer, and the transformer to the pc. Then go to the pc and change its ipv4 to 192.168.2.10. To see if we did it good we have to open a chrome console and put the ip of the reader with port 3161, i.e.:http://192.168.2.151:3161, and access to the reader with the user and password "admin".
Then, the information that we actually want to parse is the one in the inventory part of the url: http://192.168.2.151:3161/devices/AdvanReader-m4-150/inventory. There we have the structure to parse. To extract information, we need to have all connected and the reader active.

Issues and Solutions

We had some difficulties solving the parsing with AsyncTask, as it's a different thread, and the main thread has to wait to the secondary thread to have the information, because if not, the list of items is empty and we have an error. We can check the status of the thread calling the function AsyncTask.Status(), that returns us FINISHED, PENDING OR RUNNING.

We also had problems running the emulator that we were given, because the localhost of the emulator is not the same localhost as the one in our computer. But as it was a trivial problem (because we didn't need the emulator, it was not useful for us) we decided to continue parsing the real information. In order to simulate a real behaviour for our project, we downloaded the xml file from the reader simulator given in the labs and we uploaded it to a web server that was easier to access just as a normal website. Once we were parsing that xml, we were able to modify it on real time and make the necessary changes on the xml to simulate tags going from one antenna to another.

Finally, we also got problems with the final implementation because we wanted to do a 2-lane simulation to re-create a shooting competition but we couldn't find a reader with 4 working antennas so we could only make the demo with one lane, although the framework can work for multiple lanes.

Installation and running

To make it work is needed one reader and four antennas. The idea was to use two lanes in which each one uses two antennas, where one antenna is located in the target to indicate if the target has been hit (ports 1 and 2) and the other where all the tennis balls are to indicate that those balls are ready to use (ports 3 and 4). In the code we have setted that the ports one and three are for the first lane and the ports two and four for the second lane (those can be changed if some port isn't working correctly). Every lane has its own balls so the balls of the first lane cannot be used in the second and vice versa (first lane RFID numbers from 1 to 5, second lane RFID numbers from 6 to 10).

To make it run we used the emulator with the device Pixel 2 API 28 and with Android 9.