Institute Mines-Télécom
Telecom SudParis
CSC 7346: Advanced Software Engineering
Prof. Paul Gibson

Project: The 15-Puzzle

Jorge Adolfo González
Kenny Alvizuris

February 16th, 2018

15-puzzle

Architecture

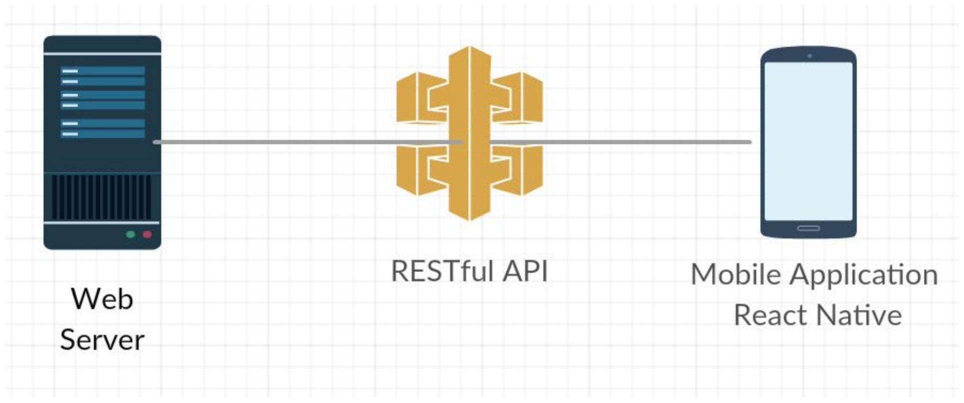


Figure 1: The Architecture

In Figure 1, we show the architecture for the project, consists in three key services.

- Web Server
- RESTful API
- Mobile Application (React Native)

Technologies

The Web service is based in Nodejs, an engine to execute javascript code in the server side, using ExpressJS which is a web application framework with many features and flexible customization.

The Application is based in React Native, which allows developers to build mobile applications using only Javascript, using the same design as React. Also, the most useful feature is that we can create cross-platform mobile applications with the same code. Therefore, our application is compatible with iOS and Android.

The Communication between the application and the web service is done via a RESTful API, written in NodeJS in the same project for the webservice in a folder named “rest-server”.

Coding Standard

To keep the same coding standard, we used ESlint to do a static check of the source code, and keep the same standard between the developers, also to comply with known javascript coding standards for better readability.

Deployment Tools

To make the project available to anyone, we used Digital Ocean as the cloud hosting, due to its accessible prices and its really straightforward to use.

Portability
To avoid the “it works on my machine” problem once and for all, we used Docker as the platform to generate builds, ship, and run the api, that allows us to without worrying about the deployment time and also to keep the api or any application isolated.

Description of Fitness Functions

We have 4 proposals for fitness functions. In order to use them, we had to normalize them from 0 to 1.

1. **Fitness Terrible**: This approach is very simple, and was one of the first approaches. If the board is resolved, then it returns 0. If it’s not resolved returns 1. This is the one that was worst in performance, we developed this algorithm to realize the importance of create a more robust fitness function that can return a wider range of values. Therefore, that’s where its name came from.

2. **Is In Place:** This approach is more complex than the first one. For this we verify the position of each number in the board, if the number is in its proper place we don’t add nothing to the final sum, if the number is an incorrect position, then we add 1 to the final sum.
   The final sum returned could be 15, being the worst case scenario since all the pieces are in an incorrect place, a number 0 would be the smallest one, confirming that the board is resolved.
   We tried to improve this fitness function by taking into account the blank space, but we noticed for some cases it helped but for the majority of complex boards it resulted in a loop or non possible way to resolve the board.

3. **Distance**: This model consists in to know how far away is each number from its proper location, and the return value is the sum of the distance of each number. With this function we have a wider range of values to normalize.

4. **Human**: This fitness function is based in the Manhattan approach, to model how a real human would resolve the puzzle, we added a weight to the distance of each number from its proper location and depending on which tile is the one being analyzed.

Work Division

Jorge González: I worked in the mobile application in React Native, creating the board in the client side, the animations of the board, the abstraction of the board, the shuffle function.

On the server side I worked on the fitness function based on the Manhattan solution, and I added a weight to know how a real human would resolve the puzzle, and to normalize all the fitness functions so we can combine all of our fitness functions to increase the accuracy.

Kenny Alvizuris: I worked in the RESTful API. There are two routes used in the project: one for resolving the puzzle and the other one to know the status of the server.

/api is the route to know the status of the server, knowing the health of server helps to handle errors in the client side and provide useful information to the user.

/api/solve is the main route in which the application sends a matrix (array of arrays) to the server, with this data, it builds a board, and start resolving the puzzle combining the fitness functions.

Screenshots

Figure 1 shows the board with the mobile version of the application, after clicking shuffle, this is how the board will look like.

Also is worth to mention, that the green squares, is to show the numbers that are in their correct position in the board.

Figure 1

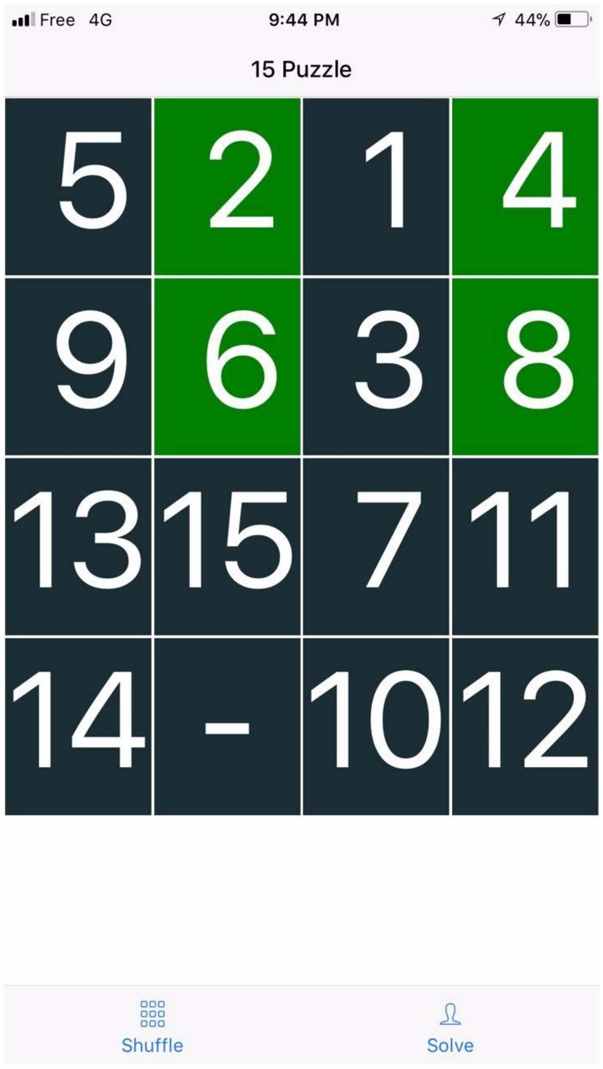


Figure 2 shows the screen shown to the user when you click “solve”.

Figure 2

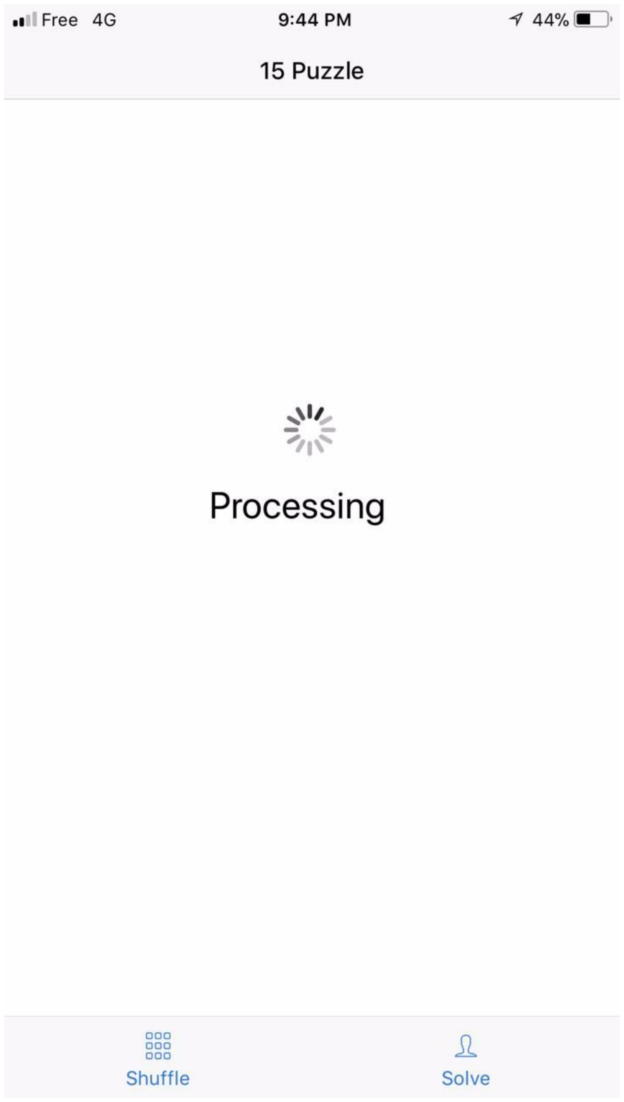
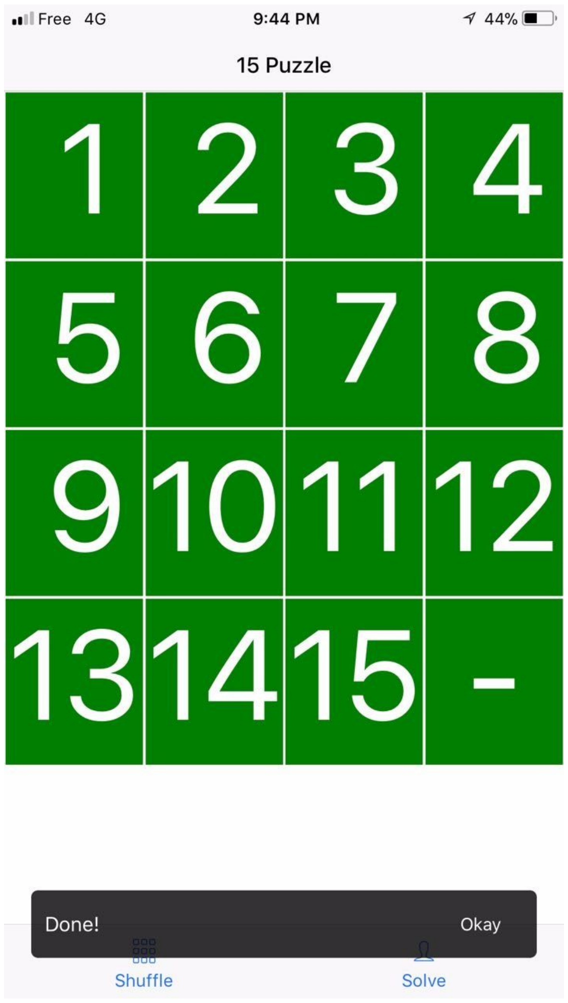


Figure 3 shows the board solved, after solving the board using our algorithms and fitness functions.

Figure 3


Free 4G
9:44 PM
44%
15 Puzzle
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
-
Done!
Okay
Shuffle
Solve