La Caza del Tesoro

Grupo K:
Tegshigzugder Otgonbayar
Tim Reiprich
Torres Ruiz Daniel Rafael
Gómez Baco José
Yepez Chavez Leticia Denise

January 26, 2020

Contents

1 Introduction

In the project of the course Cloud Application Development we were to develop a web application for a Treasure Hunt Game. By the given requirements we were to use different technologies of ...

2 General description

* deploy over a PaaS cloud
* https://github.com/DRTorresRuiz/LaCazaDelTesoro

2.1 Programming languages and frameworks

- project created using the Django framwork because Python every group member knows this language and the framework is supposed to be beginner friendly
- Python 3.7 and Django 2.2.8
- for certain tasks like the interaction with Google Maps JavaScript had to be used as there is no explicit Python API
- application deployed using Heroku (easy to handle, provides possibly useful add-ons)
- Bootstrap 4.1 and Jquery 3.4.1
- Djongo 1.2.36 to join MongoDB and Django framework
- Django-geoposition to manage coordinates

2.2 Treasure Hunt

The Treasure Hunt is a competition game in which a group of participants. They must find a series of treasures scattered throughout a play area. The location of each treasure will come indicated by a clue that suggests where it is hidden. The first winner will win the game. So the main goal of the game is to locate all the treasures.

3 Components

The Django project is split into multiple apps. Each app handles one component of the whole project. This is done to get a clear structure of the project and to simplify parallel work. Each of the following sections will explain the functionality and implementation of one specific app.

3.1 Registration

- handles user login, storing of user data and logout
- functionality provided by django-social-auth
- only logout is done using the django standard logout as earlier mentioned module uses the same models
- uses OAuth 2.0 to enable interaction with social network APIs → we used Google+ API
- module redirects new user to Google+ Login homepage, stores known users in database and provides our homepage with needed information about the user
- API can be configured using the Google developer console
- login_required (requires user to log in to access a certain view) and user.is_authenticated (possibility to check if user is logged in) are used in different apps of the project

3.1.1 Homepage

- contains initial view of our website
- shows overview of current and finished games etc.
- provides access to detail views of games and the possibility to log out
- can only be accessed if user is logged in to a Google account, otherwise user is redirected to registration app (login_required before view)

3.2 Chat

- contains functionality of interaction between users in the same game
- add descrption how chat works and why this implementation was chosen
- backend uses Redis to transmit the messages
- locally a docker container providing a Redis server was used but couldn't be launched on Heroku
- Heroku provied add-on "Heroku Redis" which was used to launch a Redis instance in the cloud to enable transmitting messages without requiring a local Redis server

3.3 Game

- provides all functionality in relation to a single game
- the functionality developed in this section can be divided into the following:
  - game relationship
  - game creation
  - game supervision
  - game play
  - administration

3.3.1 Game relationship

- provides access to aplication games
- allow to register in a game a play
- there are three lists with games: in which the user is registered, all games and games created by the user

3.3.2 Game creation

- all users can create games with treasures
- the game is created indicating the name and the play area
- to create a treasure the user must set a name, clue, solution coordinates and an optional image

3.3.3 Game supervision

- only the creator of game or administrator can see the information about the game status
- show treasures coordinates with markers in a map, the game status and users who have found each of the treasures
- the creator can reset a game, deleted all treasure findings data recorded.

3.3.4 Game play

- the registered user can access to play game if the game are not finished
- the user coordinates are retrieved from the browser and in case it is not possible the user would be allowed to select the coordinates by selecting on the map
- the user must upload a image to prove it

- when a user finds all the treasures, the game ends and other users cannot find more treasures

3.3.5 administration

- administrator users can see the status of all games

4 Database

Below is the diagram of the database used in the application:

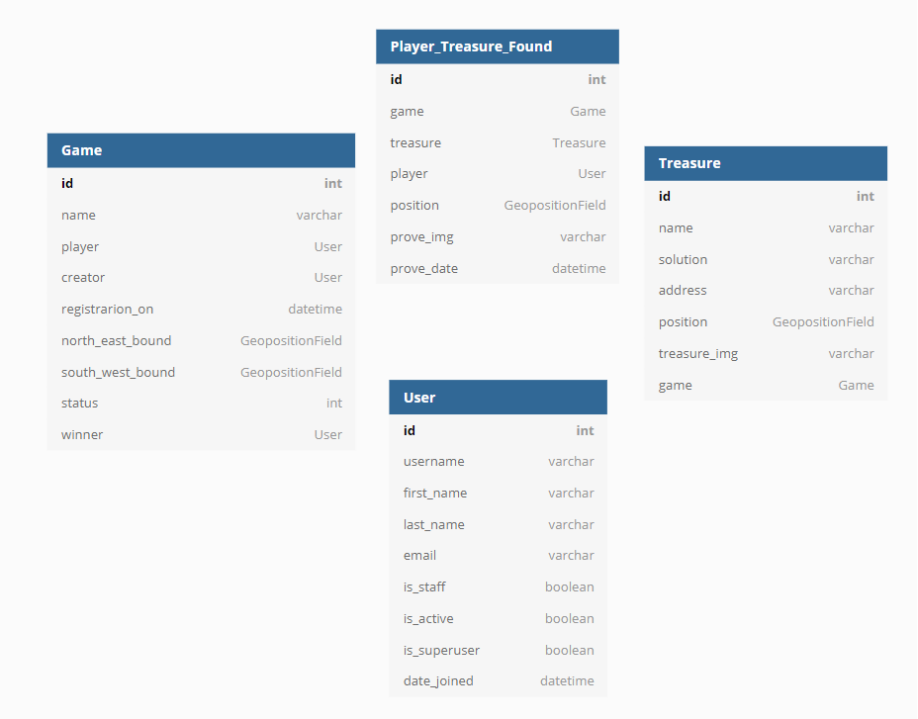

Figure 1: Database.

We use user entity provided by Django framework and others entities that are not shown as they are typical of the framework

5 User manual

- required setup (?)
- explanation of user interface

6 Limitations

- limitations of our project
- difficulties encountered
- possible improvements in the future

7 Summary

Through this project we were able to
    It is very interesting to study a