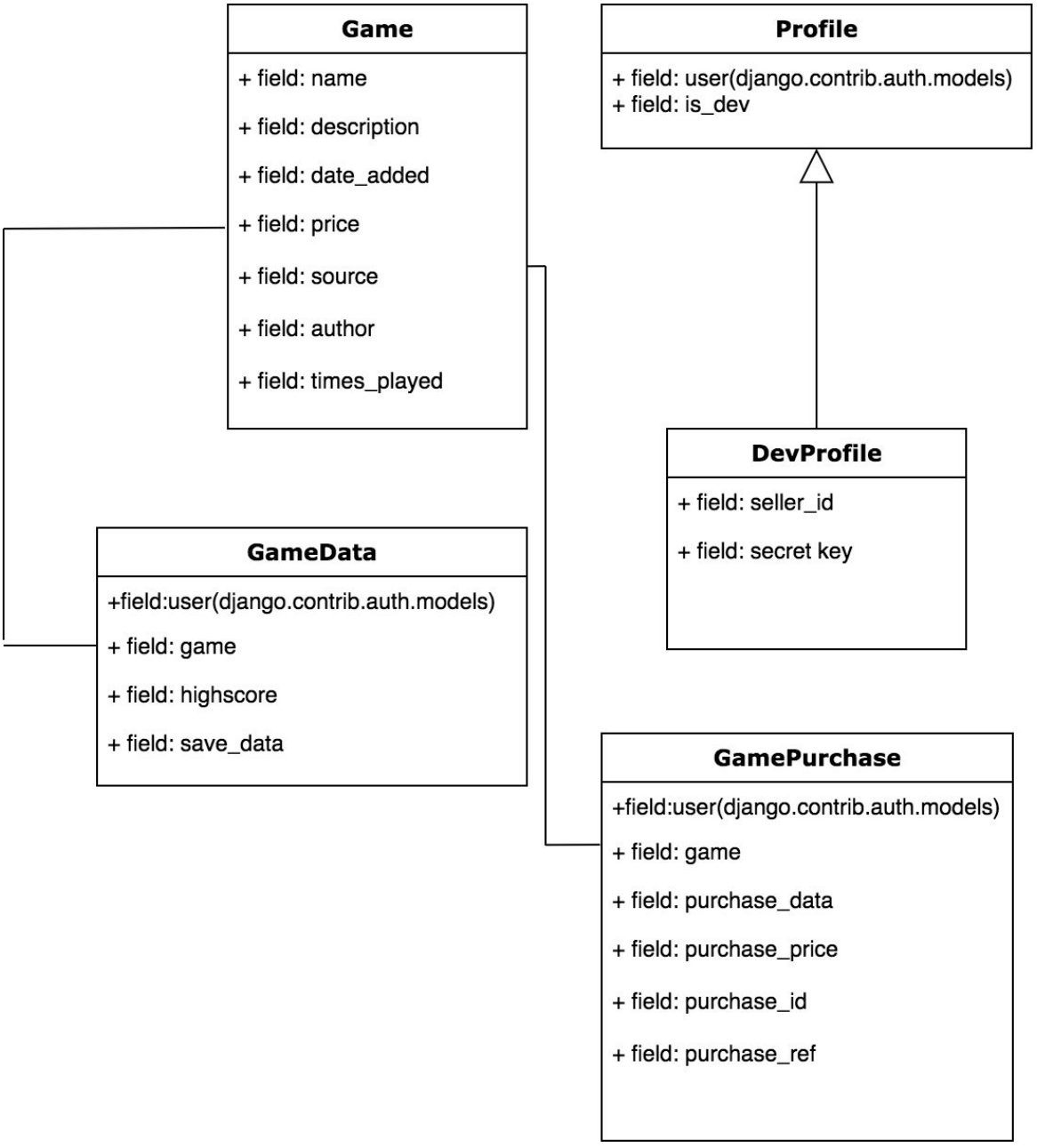
1. Team Members

Sergei Kaukiainen 586773
Aapo Linjama 587895
Oskari Mäkinen 477510

2. Features

Database schema:



Authentication 200/200:

- Authentication works as required, and account activation is done through a verification email through the console backend.

Basic player functionalities 250/300:

- Buying and browsing games works.
- Only players who have bought the game or are the game's creator are allowed to play
- Finding games could be improved by adding search and filtering functionalities

Basic developer functionalities 200/200:

All of the requirements below are fulfilled

- Add a game (URL) and set price for that game and manage that game (remove, modify)
- Basic game inventory and sales statistics (how many of the developers' games have been bought and when)
- Security restrictions, e.g. developers are only allowed to modify/add/etc. their own games, developer can only add games to their own inventory, etc.

Game/service interaction 200/200:

- The messaging system between game/service works as specified in project info.

Quality of Work 90/100:

- Code is mostly of good quality and well organized. Some styling in templates could have been moved to a separate CSS.
- Django framework and its features are used extensively and we've tried to take the cleanest approach possible to each implemented feature
- Styling could be improved infinitely, but our site is clear and logically structured

More Features

Save/load and resolution feature 100/100:

- The service supports saving and loading for games with the simple message protocol described in Game Developer Information

3rd party login 100/100

- You can register as a player by signing in with google account

RESTful API (100/100)

- Django REST framework was used for building the API. Framework provided many basic features for example tools for serialization were excellent and the framework communicates straightforwardly with django ORM.
- We implemented serializers for four models: GameData, Game, Profile, and User. For example highscores for games can be obtained through API (from GameData model)
- Authentication was also implemented to API, so that user and profile information was visible only to admins. Also only developers and admins can add new games through API.
- There were some problems concerning User model, which was solved by choosing the right class from which the serializer classes inherited features. All requirements were fulfilled.

Own game (75/100 points)

- Own game was implemented and it communicates with the platform. Only high scores were obtained in the platform (submit), therefore all requirements were not fulfilled.
- Own game is snake game and the score is measured from the size of snake

Mobile Friendly (25/50 points)

- Platform works quite well in mobile environment

3. Workload

Oskari: Authentication, basic player functionalities, game/service interaction, API, payment
Aapo: Game/service interaction, save/load and resolution feature, API, own game
Sergei: Email verification, highscores, Heroku deployment, 3rd party login

In reality everybody did something or helped in almost every feature so this is just a rough frame of workload.

4. Instructions

When testing our app locally:

1. Firstly install these:

    - pip install six
    - pip install django-allauth
    - pip install djangorestframework
    - Python 3.6 or newer
    - Django 3.0.2

2. Create admin (superuser) for app and add necessary settings for 3rd party login

    - Note: For the deployed app use the domain instead of localhost
    - In the project root folder is a text file called gmail.txt where you will find Client ID and Client Secret -key. You need to add these to admin page in order 3rd party login (gmail login) to work.
    - In admin page first go to sites -> click example.com and change Domain name to “localhost:8000” and Display name to whatever you want (for example “Gmail login”). Save changes.
    - Then in admin home page go to Social applications -> “Add social application”. New page will open after this. Choose from Provider drop menu “Google”, name application as you want and from gmail.txt file add Client id and Secret key. Lastly in Sites, move “localhost:8000” from Available sites to Chosen sites. Save changes and you are good to go with the app.
    - In the admin page you can also add a profile for your superuser (player or developer). You need to create a devProfile if you want to add games and receive payments.
    - To add our test game to the service use https://localhost:8000/play/test_game2/ as the source url

3. In the web app when you create a new account you will need to verify your account via a link that is sent to your email. Since we are using Django’s email backend, you will find this link in the console. You can also activate a user’s account in the admin page.

When testing in Heroku:

Link to Heroku: https://mostawesomegames.herokuapp.com/home/

For testing purposes we have created one superuser and one developer account to which we added given example game. Login infos:

- Superuser:
    - Username: admin
    - Password: salasana1234

- Developer account
    - Username: dev
    - Password: salasana1234
- If you want to add additional developers to the deployed app, create a player user and then create a DevProfile and Profile for the user through django admin. (Since the verification email is sent to console backend)