Habit app backend

REST API for creating habits and tracking them by users

Table of contents

Explanation of framework choice

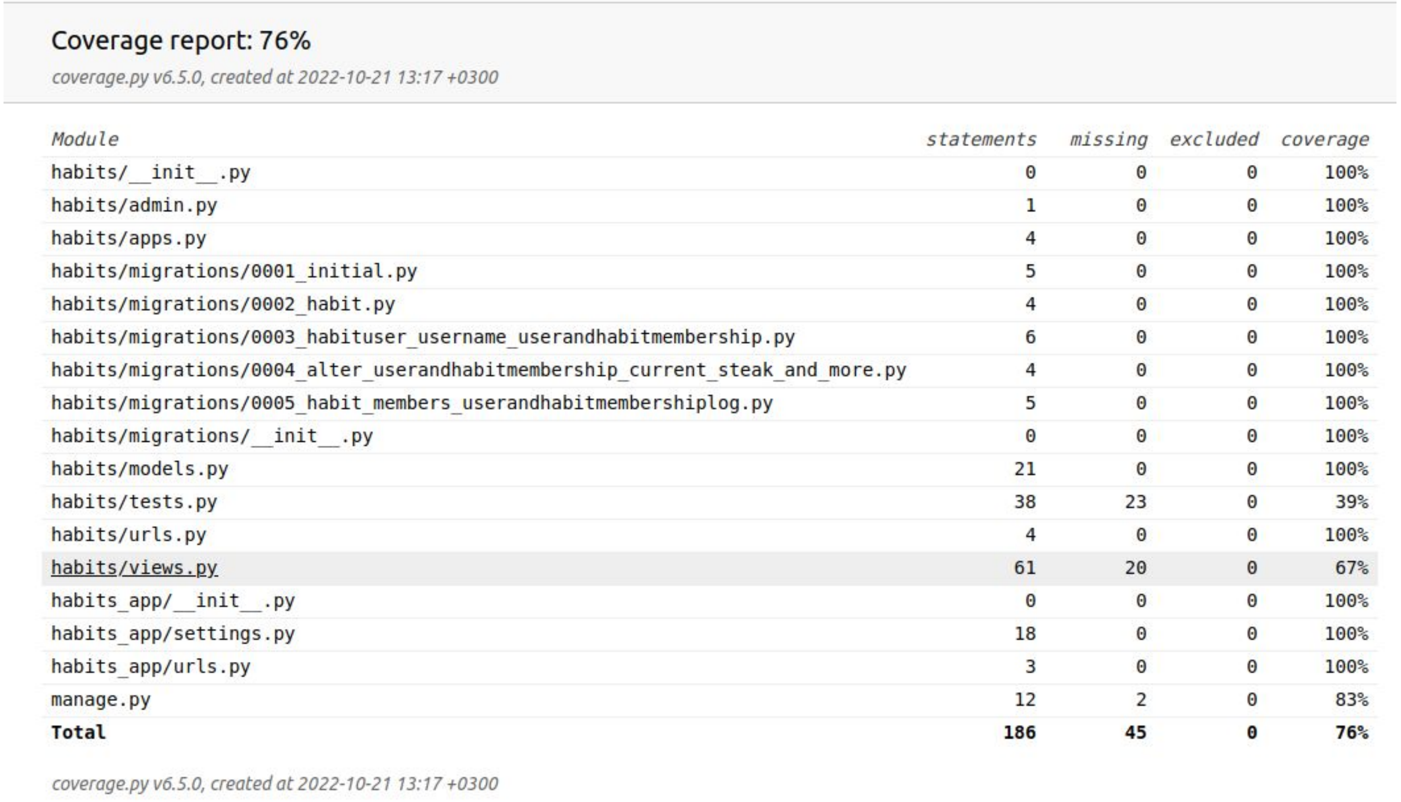
To implement this functionality, I decided to choose the django framework. The main reasons were the following:

1) The existence of ORM(object-relational mapping) for different databases. This makes it possible to select any database and switch between them later. It also makes it possible to represent the tables in the database as classes in python.
2) Ability to create an admin panel for the application out of the box. This can be useful if we want to view or change the data of our application through the web interface.
3) Huge popularity of this framework.

Tests

For test coverage, I used the standard utilities of the Django framework.Also, to measure the percentage of code coverage by tests, I used the *coverage* utility. You can install it with this command: *$ pip install coverage*. On the next page you can see detailed test coverage report.

Tests coverage report



Coverage report: 76%

coverage.py v6.5.0, created at 2022-10-21 13:17 +0300

Module	statements	missing	excluded	coverage
habits/__init__.py	0	0	0	100%
habits/admin.py	1	0	0	100%
habits/apps.py	4	0	0	100%
habits/migrations/0001_initial.py	5	0	0	100%
habits/migrations/0002_habit.py	4	0	0	100%
habits/migrations/0003_habituser_username_userandhabitmembership.py	6	0	0	100%
habits/migrations/0004_alter_userandhabitmembership_current_steak_and_more.py	4	0	0	100%
habits/migrations/0005_habit_members_userandhabitmembershiplog.py	5	0	0	100%
habits/migrations/__init__.py	0	0	0	100%
habits/models.py	21	0	0	100%
habits/tests.py	38	23	0	39%
habits/urls.py	4	0	0	100%
habits/views.py	61	20	0	67%
habits_app/__init__.py	0	0	0	100%
habits_app/settings.py	18	0	0	100%
habits_app/urls.py	3	0	0	100%
manage.py	12	2	0	83%
Total	**186**	**45**	**0**	**76%**

coverage.py v6.5.0, created at 2022-10-21 13:17 +0300

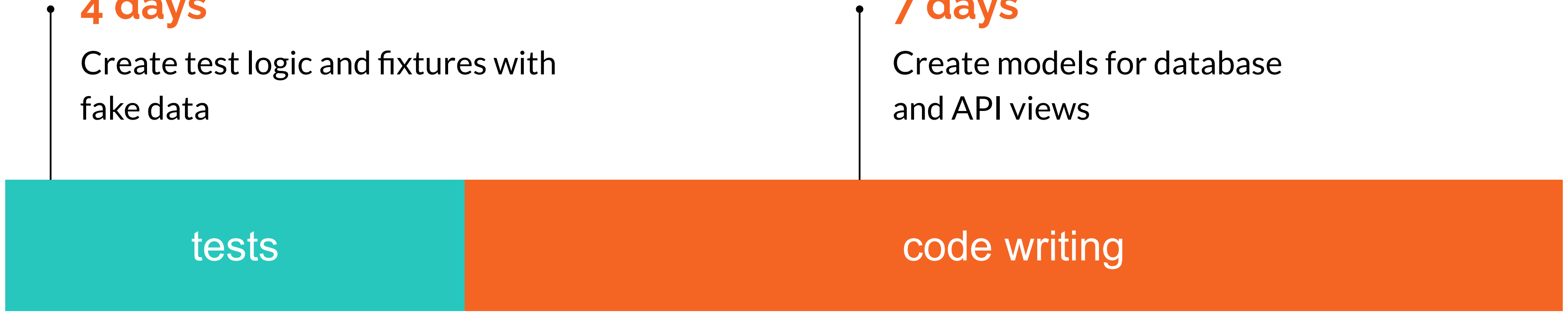
Stages of work

4 days

Create test logic and fixtures with fake data

7 days

Create models for database and API views



For this project i decide to try test-driven development. So, first I wrote all the tests that will test all the functionality. And only after that the code itself.

Description of business logic

In our service, you can register a **user**, for this you need to specify an email.

In addition, you can register any **habit** in our service. A habit consists of a name, a description, and a frequency type (daily, weekly, monthly, yearly)
In our service, you can create a **habit track**. Habit track consists of a link to the user, habits, longest streak. current streak.
We also create a **habit log** in our service. It consists of a link to a habit track and the time when the task was completed.
In addition, we decided to add an **achievement** entity to our project. Achievement consists of a name, description, link to a habit and rules of assign achievement to users.