МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Звіт

з розробки п. 5-8 курсової роботи

На тему “додаток, який створює сильні та безпечні паролі”

Студента групи ТВ-33

Домарацького Дмитра Олександровича

Посилання на github-репозиторій:

https://github.com/dmtrDO/WebCourse

Київ 2025

Розділ 5. Розробка механізмів отримання та оновлення даних;

1. Реєстрація користувача. Після натискання кнопки "Зареєструватися", дані з форми надсилаються методом POST на сервер за шляхом /registration. У тілі запиту передаються дані користувача: електронна пошта, пароль. Django обробляє ці дані через відповідне view, перевіряє їх на коректність і створює нового користувача. У випадку успішної реєстрації користувач перенаправляється на сторінку входу. Якщо введені дані некоректні (наприклад, пароль занадто короткий), користувачу повертається та сама форма з повідомленням про помилку, причому помилки, пов’язані з БД, обробляються у views, а помилки, пов’язані з довжиною паролів, або невірним форматом електронної пошти, або невірними символами при заповненні паролю і т.д., то такі помилки оброблюються через javascript.

2. Авторизація користувача. Форма входу надсилає дані методом POST на шлях /login. У формі передаються пошта і пароль. Django перевіряє ці дані аналогічно, як при реєстрації, і, якщо вони правильні — виконує вхід користувача, після чого його перенаправляють на головну сторінку застосунку.

3. Генерація паролів. При натисканні на кнопку «Згенерувати» на сервер надсилається POST-запит за тим же шляхом, з якого він надсилається, попередньо обробивши вхідні налаштування через сесії та передавши ці дані в БД.

4. Перегляд історії. Для отримання історії збережених паролів реалізовано view, який при GET-запиті до URL /history витягує всі записи паролів поточного користувача з бази даних та передає їх у шаблон для відображення. Таким чином реалізується механізм отримання даних.

Розділ 6. Розробка бази даних для збереження та отримання інформації;

У розробленому вебзастосунку використовується реляційна база даних, побудована за допомогою ORM (Object-Relational Mapping), яку надає фреймворк Django. Для зберігання та обробки даних обрано СКБД SQLite, оскільки вона:

- є вбудованою, не потребує окремої установки;
- забезпечує достатній рівень функціональності;

Схема (в більшості IDE чи програмах для роботи з базами даних схеми генеруються автоматично на основі вже створених таблиць) (рисунок 1). Тут для генерації схеми БД було використано команду в терміналі PyCharm:

```
python manage.py graph_models main -o diagram.svg
```

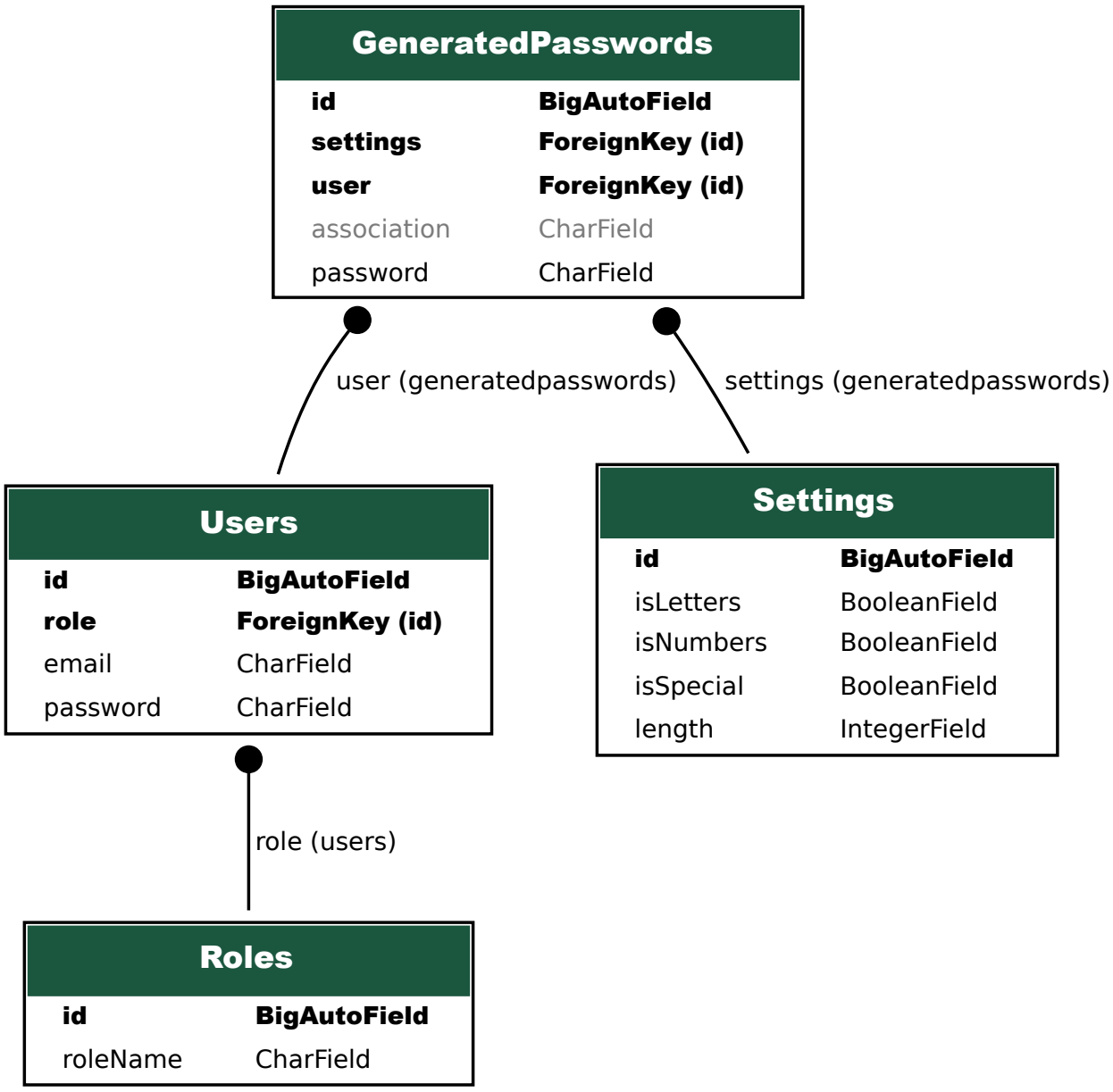

Рисунок 1 - схема БД

Поле `email` у таблиці `User` має унікальний індекс, що забезпечує швидкий пошук і перевірку унікальності. Зовнішні ключі (`user`, `settings`, `role`) автоматично індексуються Django, що забезпечує ефективність при фільтрації.

Розділ 7. Інтеграція сторонніх сервісів в систему;

У рамках реалізації даного вебзастосунку використовується сторонній хмарний сервіс PythonAnywhere, який забезпечує зручне середовище для розгортання та публічного доступу до python-додатків. Це дозволяє швидко опублікувати проєкт без потреби в оренді власного сервера.

Інтеграція з даним сервісом не потребує додаткового API, оскільки основна взаємодія з ним здійснюється через браузер або термінал: спочатку потрібно завантажити джанго-проєкт (максимальна вага — 512 Мб, треба стиснути проєкт у архів, наприклад, zip, і закинути його у сервіс у розділ Files, оскільки просто так завантажити проєкт не вийде, потім його треба розархівувати через термінал bash console, вбудований у pythonanywhere командою unzip *.zip), перейти у розділ Web, у підрозділі Code вказати абсолютний шлях до проєкту, у підрозділі Static files вказати шлях до директорії, в якій знаходяться статичні файли, редагувати конфігураційний WSGI-файл так, щоб в ньому були налаштування тільки для django і коректні шляхи до відповідних файлів і папок, змінити змінну у файлі settings.py для того, щоб надати дозвіл хостам користуватись цим додатком ALLOWED_HOSTS = [*]; і перезавантажити додаток у розділі Web→Reload.

Існують певні недоліки використання цього сервісу, які відрізняються від платного хостингу: по-перше, можливо використовувати тільки один додаток, тобто одночасно вивантажити на сервер два веб-додатка не вийде, по-друге, у розділі Web→Best before date потрібно час від часу оновлювати додаток, оскільки, якщо цього не робити протягом трьох місяців, то сайт перестане працювати на сервері, але у цій роботи ці недоліки не відчуваються.

Розділ 8. Створення серверної архітектури додатку;

Для розробки серверної частини обрано монолітну архітектуру, що дозволяє спростити розгортання та підтримку додатку на етапі розробки, а також зменшити складність управління додатком.

Основні модулі системи:

1. Модуль користувачів (Users). Модуль користувачів відповідає за управління даними користувачів, включаючи реєстрацію, авторизацію та зберігання інформації про користувачів, таких як електронна пошта, пароль тощо. Модуль взаємодіє з базою даних для збереження цих даних і використовується іншими модулями для аутентифікації та авторизації користувачів в системі.
2. Модуль налаштувань (Settings). Модуль налаштувань відповідає за параметри генерації паролів, зокрема довжину паролю, наявність цифр, літер та спеціальних символів. Він зберігає ці налаштування в базі даних і взаємодіє з іншими модулями, щоб забезпечити передачу налаштувань при генерації паролів.
3. Модуль генерації паролів (GeneratedPasswords). Цей модуль відповідає за безпечну генерацію паролів згідно з параметрами, вказаними в модулі налаштувань. Згенеровані паролі зберігаються в базі даних і асоціюються з конкретними користувачами. Модуль також надає API для доступу до згенерованих паролів.

Схема (рисунок 2):

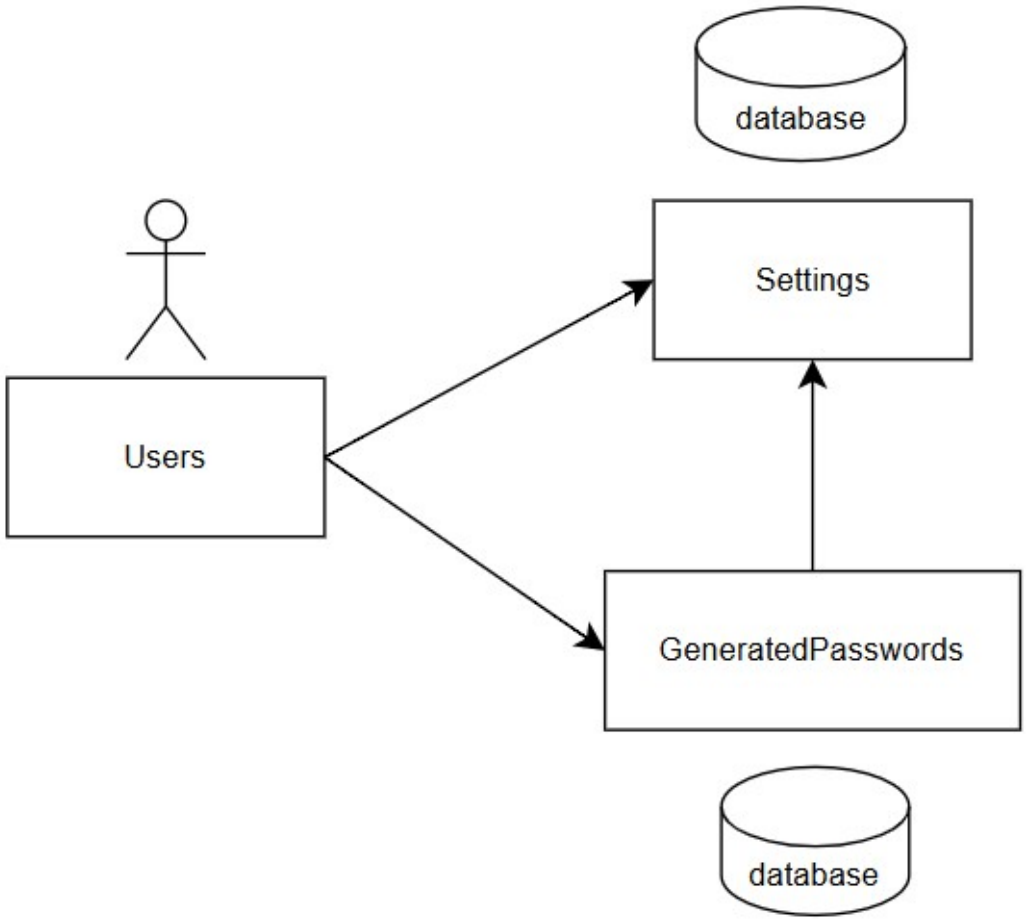

Рисунок 2 — схема монолітної архітектури