рамки Берлінської класифікації, а отже жанр потребує уточнення ключових характеристик для ігор такого типу.

Список використаних джерел та літератури

1. The making of: rogue. *https://web.archive.org*. URL: https://web.archive.org/web/20121018225934/http://www.edge-online.com/features/making-rogue/ (дата звернення: 07.11.2023).
2. Procedural dungeons of doom: the making of rogue – chapter 1. *https://episodiccontentmag.com*. URL: https://episodiccontentmag.com/2016/06/03/rogue_chapter1/ (дата звернення: 07.11.2023).
3. Berlin interpretation. *https://www.roguebasin.com*. URL: https://www.roguebasin.com/index.php/Berlin_Interpretation (дата звернення: 07.11.2023).
4. Your game is not a roguelike. *https://vildravn.dev*. URL: https://vildravn.dev/posts/your-game-is-not-a-roguelike/ (дата звернення: 07.11.2023).

Махенько Ярослав,
асистент кафедри комп'ютерних наук та інформаційних технологій
Прухницький Віталій,
асистент кафедри комп'ютерних наук та інформаційних технологій,
Стельмашенко Яніна,
здобувачка другого (магістерського) рівня вищої освіти
фізико-математичного факультету,
Житомирський державний університет імені Івана Франка,
м. Житомир, Україна

ПРОЕКТУВАННЯ БАЗИ ДАНИХ ЗАСОБАМИ ФРЕЙМВОРКУ DJANGO

В сучасних реаліях не можливо уявити сайт або веб-додаток, який не зберігає дані для подальшої обробки. Саме тому важливо використовувати базу даних для зручного керування інформацією, її зберігання та обробки для розширення можливостей веб-сайту.

Django - це фреймворк веб-розробки на Python, який забезпечує потужні засоби для роботи з базами даних. Django має власну мову моделювання, яка дозволяє розробникам легко створювати структури баз даних для своїх додатків. Тому, за мету поставлено розгляд деяких поширених прийомів та рекомендацій, які допоможуть у створенні ефективних та масштабованих баз даних.

Для початку варто розглянути перелік баз даних, який підтримує фреймворк Django:

- PostgreSQL – об'єктно-реляційна система керування базами даних, не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення.
- MariaDB – реляційна система керування базами даних, створена на початку 2009 як відгалуження (форк) MySQL. Поштовхом для створення даної системи

стала невпевненість спільноти розробників та користувачів у долі і ліцензії MySQL після її придбання в Oracle.

● MySQL – вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» з метою збільшення швидкості обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

● Oracle – об'єктно-реляційна система керування базами даних від Oracle Corporation.

● SQLite – полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92.

У Django бази даних представляють класами моделей. Модель – це опис структури таблиці в базі даних. Вона визначає імена полів таблиці, типи даних полів і їхні обмеження. Для створення моделі використовується клас Model з пакету django.db.models. До прикладу, так виглядатиме створення моделі книжкового магазину (рис.1):

```
from django.db import models

class Shop(models.Model):
    name = models.CharField('Назва', max_length=200)
    address = models.TextField('Адреса',)
    opening_time = models.TimeField('Час відкриття')
    closing_time = models.TimeField('Час закриття')

    def __str__():
        return f'{self.name} / {self.opening_time}-{self.closing_time}'

    class Meta:
        verbose_name = 'Магазин'
        verbose_name_plural = 'Магазини'
```

Рисунок 1. Створення моделі книжкового магазину

Після цього, варто імпортувати django.db.models та створити модель Shop яка має 4 поля:

● *name* – назва магазину, поле типу CharField, яке може містити в собі невеликий текст, обмежуємо розмір поля в 200 символів.

● *address* – адреса магазину, поле типу TextField, яке може містити в собі великий текст.

● *opening_time* та *closing_time* – час відкриття і закриття магазину, поля типу TimeField, які містять в собі екземпляр datetime.time, а саме – час.

Для подальшої роботи з сайтом на цьому етапі потрібно створити модель «Жанр» (рис. 2), що має містити лише одне поле типу CharField – назва жанру:

```
class Genre(models.Model):
    name = models.CharField('Назва', max_length=200)

    def __str__():
        return f'{self.name}'

    class Meta:
        verbose_name = 'Жанр'
        verbose_name_plural = 'Жанри'
```

Рисунок 2. Створення моделі «Жанр».

Наступним етапом є створення останньої моделі – «Book» (рис. 3). Вона має містити такі поля, як:

- *name* – назва книги, поле типу CharField, яке може містити в собі невеликий текст, обмежуємо розмір поля в 200 символів.
- *genre* – жанр книги, поле типу ForeignKey, яке посилається на модель Genre.
- *quantity* – кількість книг, поле типу IntergerField, яке може містити в собі ціле число.
- *price* – вартість книги, поле типу FloatField, яке може містити в собі будь яке число.

```
class Book(models.Model):
    name = models.CharField('Назва', max_length=200)
    genre = models.ForeignKey(Genre, models.PROTECT, verbose_name='Жанр')
    quantity = models.IntergerField('Кількість')
    price = models.FloatField('Вартість')

    def __str__():
        return f'{self.name} / {self.price}'

    class Meta:
        verbose_name = 'Книга'
        verbose_name_plural = 'Книги'
```

Рисунок 3. Створення моделі «Book».

Зручність проектування бази даних за допомогою моделей полягає в тому, що код є універсальним для будь якої з поданих баз даних.

Django використовує міграції для створення і оновлення баз даних. Міграція - це файл, який описує зміни, які потрібно внести в базу даних.

Для створення міграції використовується команда makemigrations. Наприклад, команда *python manage.py makemigrations* створить міграцію для створення таблиць Shop, Genre та Book у базі даних. Для внесення змін описаних у міграції, Django використовує команду migrate. Наприклад, команда *python manage.py migrate* застосує міграцію та створить таблиці в базі даних.

При проектуванні бази даних засобами Django важливо дотримуватись певних правил:

● Використовувати правильні типи полів. Обирати типи полів, які відповідають типу даних, які ви зберігаєте.

● Використовувати обмеження. Обмеження допомагають запобігти помилкам і забезпечити цілісність даних. Дуже важливо правильно проводити валідацію даних, встановлювати максимальні та мінімальні значення для числових полів, та розміри для текстових.

● Створювати зв'язки. Модельні зв'язки допомагають зв'язувати декілька таблиць, як на нашому прикладі, ми зв'язали таблицю з переліком книг з таблицею жанрів.

Проектування баз даних є важливим етапом розробки будь-якого веб-сайту або веб-додатку. Django надає потужні засоби для проектування і створення баз даних. Використовуючи ці засоби, можна створити ефективні та масштабовані бази даних, які будуть відповідати потребам сучасних тенденцій.

Список використаних джерел та літератури

1. Databases. Django documentation. Django: веб-сайт. URL: https://docs.djangoproject.com/en/4.2/ref/databases/#postgresql-notes (дата звернення 23.10.2023).

2. PostgreSQL: The world's most advanced open source database: веб-сайт. URL: https://www.postgresql.org/ (дата звернення 23.10.2023)

3. MariaDB Foundation - MariaDB.org: веб-сайт. URL: https://mariadb.org/ (дата звернення 23.10.2023).

4. Oracle Ukraine| Cloud Applications and Cloud Platform: веб-сайт. URL: https://www.oracle.com/ua/ (дата звернення 23.10.2023).

Проноза Ярослав,
здобувач першого (бакалаврського) рівня вищої освіти
фізико-математичного факультету
Науковий керівник: ***Федорчук Анна,***
кандидат педагогічних наук, доцент,
доцент кафедри комп'ютерних наук та інформаційних технологій,
Житомирський державний університет імені Івана Франка,
м. Житомир, Україна

ВИКОРИСТАННЯ UX ТА UI ДИЗАЙНУ В МОБІЛЬНИХ ЗАСТОСУНКАХ

Мобільні застосунки стали невід'ємною частиною нашого життя. Ми використовуємо їх для розваг, покупок, спілкування, виконання різних робочих завдань. Однак присутність великої кількості застосунків на ринку означає, що розробникам потрібно вирізнятися з поміж конкурентів, щоб привернути увагу користувачів і зробити їх щасливими. Саме ці функції виконує UX (англ. user experience – користувацький досвід) та UI (англ. user interface – інтерфейс користувача) дизайн.

UX/UI дизайн відіграє ключову роль у розробці мобільних застосунків. Він відповідає за взаємодію користувача з продуктом і його враження від цієї