PYTHON331 HW №32
Домашнее задание

Краткое содержание задания

Студентам предстоит обновить базу данных для Django-приложения, удалив из таблицы `Cards` столбец `UserID`, который может вызывать проблемы при работе с Django. Затем необходимо интегрировать обновленную базу данных в Django-приложение, адаптировать модели под новую структуру данных, выполнить миграции и обновить представления и шаблоны для работы с новыми моделями.

Детальная пошаговая инструкция

1. **Удаление столбца `UserID` из таблицы `Cards`:**
   - Студентам предлагается удалить столбец `UserID` из таблицы `Cards` в базе данных. Это можно сделать либо через графический интерфейс SQLiteStudio, либо с использованием серии SQL команд. (Команды для удаления столбца будут предоставлены отдельно).

```sql
PRAGMA foreign_keys = 0;

-- Создание временной таблицы без столбца UserID
CREATE TABLE sqlitestudio_temp_table AS SELECT CardID, Question, Answer, CategoryID, UploadDate, Views, Favorites FROM Cards;

-- Удаление оригинальной таблицы Cards
DROP TABLE Cards;

-- Создание новой таблицы Cards без столбца UserID
CREATE TABLE Cards (
    CardID INTEGER PRIMARY KEY AUTOINCREMENT,
    Question TEXT NOT NULL,
    Answer TEXT NOT NULL,
    CategoryID INTEGER,
    UploadDate DATETIME DEFAULT (datetime('now')),
    Views INTEGER DEFAULT (0),
    Favorites INTEGER DEFAULT (0),
    FOREIGN KEY (CategoryID) REFERENCES Categories (CategoryID) ON DELETE SET NULL ON UPDATE CASCADE
);

-- Копирование данных обратно в Cards из временной таблицы
INSERT INTO Cards (CardID, Question, Answer, CategoryID, UploadDate, Views, Favorites)
SELECT CardID, Question, Answer, CategoryID, UploadDate, Views, Favorites FROM sqlitestudio_temp_table;

-- Удаление временной таблицы
DROP TABLE sqlitestudio_temp_table;

PRAGMA foreign_keys = 1;
```

2. **Подключение обновленной базы данных к Django-приложению:**
   - Опционально: Делаем ветку Git для этих работ.
   - Поместите новую базу данных рядом со старой.
   - Измените имя базы данных в файле `settings.py` вашего Django-проекта, чтобы оно соответствовало новой базе данных.
3. **Описание моделей Django:**
   - Опишите модели в Django, которые соответствуют структуре вашей новой базы данных. При описании моделей учтите, что Django имеет свои правила именования, и новая база данных может не соответствовать этим стандартам.
   - Используйте параметр `db_column` в объявлении полей модели для указания конкретного имени столбца в базе данных. Это позволяет связать поле модели с определенным столбцом в таблице БД, даже если их имена различаются.
   - В классе `Meta` каждой модели используйте параметр `db_table` для явного указания имени таблицы в базе данных, с которой должна быть связана модель.
   - Для связи "многие ко многим" между карточками и тегами используйте конструкцию `models.ManyToManyField`, указывая параметр `through` для определения промежуточной таблицы, которая управляет связью.
4. **Выполнение миграций:**
   - Сначала выполните миграцию без предварительного создания файлов миграции командой `./manage.py migrate` для создания служебных таблиц Django в базе данных.
   - Затем создайте файлы миграции с помощью `./manage.py makemigrations` и выполните их фейково, чтобы Django ORM "подумал", что миграции применены, но на самом деле структура таблиц осталась неизменной. Для этого используйте команду `./manage.py migrate --fake`.
5. **Обновление представлений и шаблонов:**
   - Поправьте представления и шаблоны в соответствии с новыми моделями, чтобы восстановить работоспособность приложения.
   - В шаблонах циклы по тегам теперь должны выглядеть как `{% for tag in card.tags.all %}`, а обращение к названию тега - как `{{ tag.name }}`.
6. **Реализация нового функционала - показ карточек по тегу:**
   - Добавьте новый маршрут для просмотра карточек по тегу: `path('tags/<int:tag_id>/', views.get_cards_by_tag, name='cards_by_tag')`.
   - Создайте новое представление `get_cards_by_tag`, которое будет извлекать все карточки, связанные с конкретным тегом, и передавать их в шаблон для отображения.
   - Сделайте теги в каталоге и в детальном представлении карточек кликабельными, чтобы пользователи могли кликнуть по тегу и увидеть все карточки, ассоциированные с этим тегом.

Важные моменты:

- **Параметр `db_column`**: Это параметр, который используется в полях модели Django для указания имени столбца в базе данных, к которому будет привязано это поле. Это особенно полезно, когда имена столбцов в базе данных не соответствуют обычным соглашениям именования Django. Например, если столбец в базе данных называется `CardID`, а вы хотите, чтобы в модели Django он имел имя `id`, вы можете использовать `db_column` для связывания этих двух:

```python
id = models.AutoField(primary_key=True, db_column='CardID')
```

- **Параметр `db_table` в классе `Meta`**: Этот параметр используется внутри класса `Meta`, объявленного в моделях Django, для указания имени таблицы базы данных, к которой модель должна быть привязана. Это необходимо, если имя таблицы в базе данных отличается от автоматически сгенерированного Django имени (которое обычно формируется из названия приложения и модели). Например:

```python
class Meta:
    db_table = 'Cards'
```

- **Связь многие ко многим через промежуточную таблицу**: При использовании `models.ManyToManyField` с параметром `through` вы явно указываете модель, которая будет использоваться в качестве промежуточной таблицы для управления связью многие ко многим. Это позволяет дополнительно настроить связь, добавив дополнительные поля в промежуточную таблицу, если это необходимо. В вашем случае, это выглядит так:

```
class CardTags(models.Model):
    card = models.ForeignKey('Card', on_delete=models.CASCADE, db_column='CardID')
    tag = models.ForeignKey('Tag', on_delete=models.CASCADE, db_column='TagID')

    class Meta:
        db_table = 'CardTags'
        unique_together = (('card', 'tag'),)
```

После внесения всех изменений и выполнения миграций, убедитесь, что ваше приложение работает корректно с обновленной структурой базы данных. Это включает в себя проверку всех измененных представлений и шаблонов, а также тестирование нового функционала для просмотра карточек по тегу.

Критерии проверки 👌

- База данных успешно обновлена, столбец `UserID` удален из таблицы `Cards`.
- Обновленная база данных успешно подключена к Django-приложению.
- Модели Django успешно адаптированы под новую структуру данных.
- Миграции успешно выполнены и применены с помощью команды `./manage.py migrate --fake`.
- Представления и шаблоны успешно обновлены в соответствии с новыми моделями.
- Новый функционал - показ карточек по тегу - работает корректно.
- Опционально. Модели подключены к админке, создан суперпользователь
- **Сдаём с БАЗОЙ ДАННЫХ!**