Техническое задание (ТЗ)

1. Введение

Система предназначена для отслеживания посещаемости студентов на лекциях с использованием NFC-меток, расположенных в каждой аудитории. Студенты смогут регистрировать своё присутствие путем сканирования NFC-метки, а администраторы смогут получать отчеты о посещаемости за определенные периоды.

---

2. Цели и задачи

- **Автоматизация учета посещаемости** студентов на лекциях.
- **Обеспечение точности данных** о присутствии и отсутствии студентов.
- **Предоставление инструментов** для генерации отчетов о посещаемости за выбранные периоды.

---

3. Структура данных

3.1. Таблица "Студенты"

- **Поля:**
  - `id` : уникальный идентификатор студента (PK).
  - `full_name` : ФИО студента.
  - `email` : электронная почта.
  - `group` : ссылка на группу (`ForeignKey` к таблице "Группы").

3.2. Таблица "Группы"

- **Поля:**
  - `id` : уникальный идентификатор группы (PK).
  - `name` : название или номер группы.

3.3. Таблица "Метки"

- **Поля:**
  - `id` : уникальный идентификатор записи метки (PK).
  - `tag_id` : уникальный код метки (ID метки из NFC).
  - `auditorium` : ссылка на аудиторию (`ForeignKey` к таблице "Аудитории").

3.4. Таблица "Аудитории"

- **Поля:**
  - `id` : уникальный идентификатор аудитории (PK).
  - `name` : название или номер аудитории.

3.5. Таблица "Лекции"

- **Поля:**
  - `id` : уникальный идентификатор лекции (PK).
  - `name` : название лекции.
  - `date` : дата проведения.
  - `start_time` : время начала.
  - `end_time` : время окончания.
  - `auditorium` : ссылка на аудиторию (`ForeignKey` к таблице "Аудитории").
  - `groups` : список групп, посещающих лекцию (`ManyToManyField` к таблице "Группы").

3.6. Таблица "Регистрации"

- **Поля:**
  - `id` : уникальный идентификатор регистрации (PK).
  - `student` : ссылка на студента (`ForeignKey` к таблице "Студенты").
  - `group` : ссылка на группу (`ForeignKey` к таблице "Группы").
  - `registration_time` : время регистрации.
  - `auditorium` : ссылка на аудиторию (`ForeignKey` к таблице "Аудитории").
  - `tag` : ссылка на метку (`ForeignKey` к таблице "Метки").
  - `lecture` : ссылка на лекцию (`ForeignKey` к таблице "Лекции").

---

4. Процессы системы

4.1. Регистрация посещаемости студентом

- Студент авторизуется в системе через приложение или веб-интерфейс.
- Сканирует NFC-метку в аудитории.
- Система получает `tag_id` и определяет аудиторию.
- На основе текущего времени и аудитории система определяет соответствующую лекцию.
- Создается запись в таблице "Регистрации" с информацией о студенте, времени регистрации, аудитории и лекции.

4.2. Генерация отчетов о посещаемости

- Администратор выбирает:
  - Промежуток дат.
  - Аудиторию.

- Система отображает список лекций, проходивших в указанной аудитории за выбранный период.
- После выбора лекции система сравнивает списки студентов групп, привязанных к лекции, с регистрациями.
- Формируется отчет со списками присутствующих и отсутствующих студентов по группам.

---

5. Пользовательские интерфейсы

5.1. Интерфейс для студентов

- **Авторизация**: возможность входа в систему по логину и паролю.
- **Сканирование метки**:
  - Экран с инструкцией по сканированию NFC-метки.
  - Подтверждение успешной регистрации после сканирования.

5.2. Интерфейс для администраторов

- **Авторизация**: вход с повышенными правами доступа.
- **Запрос отчетов**:
  - Выбор промежутка дат.
  - Выбор аудитории.
  - Отображение списка лекций.
- **Просмотр отчетов**:
  - Таблица с данными о посещаемости по группам.
  - Возможность экспорта данных в CSV или Excel.

---

6. Технические требования

6.1. Backend

- **Фреймворк**: Django с использованием Django REST Framework (DRF).
- **База данных**: PostgreSQL.
- **Аутентификация**: JWT (JSON Web Tokens) через `djangorestframework-simplejwt`.
- **API эндпоинты**:
  - Авторизация и регистрация пользователей.
  - Эндпоинт для регистрации посещаемости.
  - Эндпоинты для получения данных о лекциях, аудиториях и отчетах.

6.2. Frontend

- **Фреймворк**: Vue.js.
- **Компоненты**:
  - Страницы авторизации для студентов и администраторов.
  - Интерфейс для сканирования NFC-меток.
  - Формы для выбора параметров отчетов.
  - Отображение таблиц с отчетами.
- **HTTP-библиотека**: Axios для взаимодействия с API.

6.3. NFC-интеграция

- **Технология**: использование Web NFC API (при поддержке браузера) или разработка мобильного приложения.
- **Альтернативы**: при невозможности использовать NFC, предусмотреть сканирование QR-кодов или ввод кодов вручную.

---

7. Безопасность и права доступа

- **Роли пользователей**:
  - **Студенты**: могут регистрировать посещаемость и просматривать свою историю (опционально).
  - **Администраторы**: полный доступ к данным и возможность генерации отчетов.
- **Шифрование данных**: использование HTTPS для защиты данных при передаче.
- **Проверка данных**: валидация и санитаризация всех вводимых данных для предотвращения атак.

---

8. Дополнительные требования

- **Обработка ошибок**: информирование пользователей о неудачных операциях и причинах ошибок.
- **Логирование**: ведение журналов действий для аудита и отладки.
- **Масштабируемость**: архитектура должна поддерживать рост числа пользователей и объемов данных.

План реализации через Django REST Framework и Vue.js

Этап 1: Инициализация проекта

- **Backend**:
  - Создать новый проект Django.
  - Установить необходимые пакеты:

    ```
    pip install django djangorestframework djangorestframework-simplejwt django-cors-headers psycopg2-binary
    ```

  - Настроить базу данных PostgreSQL.

    - Настроить `settings.py` для использования DRF и JWT.

- **Frontend**:
    - Создать новый проект Vue.js:

      ```
      vue create attendance-system
      ```

    - Установить Axios и другие необходимые пакеты:

      ```
      npm install axios vue-router
      ```

Этап 2: Моделирование данных (Backend)

- **Создать модели**:
    - Определить модели согласно техническому заданию.
    - Связать модели с помощью `ForeignKey` и `ManyToManyField`.
- **Выполнить миграции**:
    - Создать и применить миграции для создания таблиц в базе данных.

Этап 3: Настройка аутентификации

- **Настроить JWT**:
    - В `settings.py` добавить настройки для `djangorestframework-simplejwt`.
- **Создать сериализаторы и представления для аутентификации**:
    - Реализовать эндпоинты для регистрации и входа пользователей.

Этап 4: Реализация API эндпоинтов

- **Сериализаторы**:
    - Создать сериализаторы для всех моделей.
- **Представления (Views)**:
    - Использовать `ModelViewSet` для стандартных операций CRUD.
    - Реализовать кастомные действия для регистрации посещаемости и генерации отчетов.
- **Маршрутизация**:
    - Использовать `DefaultRouter` для автоматической генерации URL.

Этап 5: Реализация функционала регистрации посещаемости

- **Эндпоинт для регистрации посещаемости**:
    - Принимать `tag_id` и токен пользователя.
    - Определять аудиторию и текущую лекцию.
    - Создавать запись в таблице "Регистрации".

Этап 6: Реализация генерации отчетов

- **Эндпоинт для получения лекций**:
    - Принимать параметры даты и аудитории.
    - Возвращать список лекций, соответствующих критериям.
- **Эндпоинт для генерации отчетов**:
    - Принимать идентификатор лекции.
    - Возвращать данные о присутствующих и отсутствующих студентах по группам.

Этап 7: Реализация интерфейса (Frontend)

- **Маршрутизация**:
    - Настроить `vue-router` для управления страницами.
- **Компоненты для студентов**:
    - Страница входа.
    - Страница сканирования метки.
        - Реализовать функционал сканирования NFC или альтернативы.
- **Компоненты для администраторов**:
    - Страница входа.
    - Форма запроса отчета.
    - Компонент отображения таблицы отчета.
- **Интеграция с Backend**:
    - Использовать Axios для выполнения запросов к API.
    - Настроить обработчики ответов и ошибок.

Этап 8: NFC-интеграция

- **Исследовать возможности**:
    - Проверить поддержку Web NFC API в целевых браузерах.
- **Реализация NFC-функционала**:
    - Если возможно, использовать Web NFC API для считывания меток.
    - При необходимости, разработать мобильное приложение или использовать альтернативные методы (QR-коды).

Этап 9: Тестирование

- **Backend**:
    - Написать тесты для моделей, сериализаторов и представлений.
- **Frontend**:
    - Тестировать пользовательские сценарии.
    - Проверить корректность отображения данных и работу функционала.

Этап 10: Развертывание

- **Настройка сервера**:
  - Выбрать хостинг-платформу (например, AWS, Heroku, DigitalOcean).
  - Настроить веб-сервер (Nginx) и приложение (Gunicorn).
- **Настройка базы данных**:
  - Обеспечить безопасность и резервное копирование.
- **Развертывание Frontend**:
  - Сборка приложения Vue.js для продакшена.
  - Размещение статических файлов на сервере или CDN.

Этап 11: Документация и обучение

- **Создание пользовательских инструкций**:
  - Для студентов по использованию системы.
  - Для администраторов по генерации отчетов.
- **API-документация**:
  - Использовать Swagger или DRF-документацию для описания эндпоинтов.

Этап 12: Поддержка и развитие

- **Мониторинг**:
  - Настроить системы мониторинга для отслеживания производительности и ошибок.
- **Обновления**:
  - Планировать обновления и добавление нового функционала на основе обратной связи.

Примечания

- **Обработка временных зон**: убедиться, что время регистрации и лекций учитывает временные зоны.
- **Соблюдение законов о защите данных**: при необходимости, обеспечить соответствие требованиям GDPR или локальных нормативов.
- **Масштабируемость и отказоустойчивость**: архитектура должна предусматривать возможность горизонтального и вертикального масштабирования.

Заключение

Данный план обеспечивает подробное руководство по реализации системы учета посещаемости студентов с использованием Django REST Framework и Vue.js. Он охватывает все аспекты проекта — от постановки задачи и моделирования данных до разработки, тестирования и развертывания приложения. Следуя этому плану, можно создать функциональную и надежную систему, отвечающую всем требованиям.