Практическое задание.

Цель: разработать backend (REST api) часть сервиса для автоматизации ресторана.

Необходимые знания: Понимание REST api, что такое HTTP протокол и из чего он состоит, какие бывают типы и статусы HTTP запросов, что такое Header в HTTP запросах, базовое понимание баз данных, проектирование баз данных (знание что такое уникальное значение, первичный ключ, внешний ключ, индекс), понимание как должны строится взаимодействие frontend и backend.

Инструменты, которые необходимо использовать и знать: Ubuntu 20.04+, nginx, php v8.1+, Laravel v10, PostgreSQL v12, postman, клиентское программное приложение SQL и инструмент администрирования баз данных (DBeaver, DataGrid jetbrains).

Дополнительные инструменты: shell, docker, docker compose

План выполнения (примерный)

1. Развернуть приложение локально, используя nginx. (https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-laravel-with-nginx-on-ubuntu-20-04)
    a. Поставить php, необходимые модули, создать проект Laravel
    b. Создать пустую БД в Postgres (не создавать таблицы)
    c. Настроить .env (сгенерировать ключ, прописать подключение к БД)
    d. Настроить nginx конфиг. Перемещать проект в /var/www и передавать права www-data не нужно!
2. Реализовать авторизацию
    a. Создать таблицу пользователей
    b. Создать роли и права
    c. Реализовать seeder для пользователей
    d. Реализовать логин, логаут, сброс и восстановление пароля
3. Реализовать работу с пользователями
    a. Модифицировать (если нужно) таблицу пользователей
    b. Реализовать CRUD-операции над пользователями
    c. Добавить проверку прав (Laravel Policies)
4. Реализовать работу с категориями меню
    a. Создать таблицу категорий меню
    b. Реализовать заполнение таблицы базовыми категориями (на свой выбор, например: завтрак, бар, десерты…)
    c. Реализовать CRUD-операции над категориями с проверками прав
    d. Реализовать работу с файлами/картинками (хранилище, storage). Работу с файлами вынести в сервис.
5. Реализовать работу с блюдами
    a. Создать таблицу блюд
    b. Реализовать заполнение таблицы базовыми блюдами
    c. Реализовать CRUD-операции над блюдами с проверками прав
    d. Реализовать просмотр блюд по категории
    e. Реализовать просмотр меню
6. Реализовать работу с заказами
    a. Создать таблицу заказов
    b. Реализовать CRUD-операции над заказами с проверками прав
7. Реализовать работу с отчетностью
    a. Создать таблицу для хранения отчетности
    b. Создать artisan-команду для формирования отчетов за день
    c. Реализовать возможность просмотра отчетов в соответствии с правами
8. Написать swagger к эндпоинтам (open-api v3, .yaml. Ссылка: https://editor.swagger.io/)
9. Создать тесты (доп.)
    a. Написать docker-compose.yaml файл. Необходимо поднять api и бд (можно обойтись базовым образом php-fpm)
    b. Создать набор feature тестов для каждого ендпоинта. Там, где требуется обращение к сторонним сервисам (почта), применить Mocking (Laravel -> Testing -> Mocking).
    c. Для теста artisan-команд используются unit-тесты.

При удалении сущности все связи должны также быть удалены (напр., при удалении категории, входящие в нее блюда должны быть также удалены).

В запросах index (получить список сущностей) реализовать пагинацию.

Отношения должны быть загружены до попадания в ресурсы.

Таблицы должны создаваться с помощью миграций Laravel (Laravel -> Database -> Migrations).

Все входные параметры должны проходить валидацию (Laravel -> The Basics -> Validation).

Сохранение файлов должно осуществляться с помощью фасада Storage (Laravel -> Digging Deeper -> File Storage).

Отношения между моделями должны реализовываться с помощью (Laravel -> Eloquent ORM -> Relationships).

Ссылка на документацию: https://laravel.com/docs/9.x

Описание основных сущностей.

Пользователь. Представляет собой пользователя системы, имеющего определенные права. Пользователь может авторизоваться в системе, и далее, в зависимости от набора прав, осуществлять необходимые действия над сущностями системы.

Категория меню. Сущность для классификации блюд. Запрос одновременно всех категорий будет представлять собой полное меню ресторана.

Блюдо. Представляет собой набор информации о блюде в меню. Блюдо относится к определенной категории.

Заказ. Содержит информацию о наборе блюд, стоимости и т.п. Полная информация о выполненном заказе представляет собой кассовый чек.

Можно добавлять сущности на свое усмотрение. Описанные таблицы ниже можно расширять.

Описание функционала.

Взаимодействие с пользователем происходит посредством HTTP запросов к API серверу.

Работа с пользователем.

Пользователь может иметь одну из ролей:
- Super admin (администратор системы)
- Администратор (ресторана)
- Официант

Пользователь
- имя - email (уникальное) - пароль - пинкод (уникальное) - роль

Необходимо реализовать функционал авторизации в системе. Авторизацию пользователей реализовать с помощью Laravel Sanctum (Laravel -> Security -> Authentication). Создание первого пользователя (супер-админ) должно происходить с помощью сидера (Laravel -> Database -> Seeding). Проверка авторизации должна происходить с помощью посредника (Laravel -> The Basics -> Middleware).

- Вход в систему.

Вход администратора и супер-админа осуществляется по email+пароль или пинкоду, вход официанта – возможен только по пинкоду.

- Выход из системы.
- Сброс пароля. Пользователь должен получить на почту письмо, содержащее ссылку с токеном для восстановления пароля.
- Восстановление пароля.

Необходимо реализовать CRUD-операции над пользователями.

- Создание. Все поля являются обязательными при создании (кроме id).
- Редактирование. Редактировать можно все поля, кроме пароля и пинкода
- Просмотр всех. Необходимо реализовать сортировку по имени и роли, а также поиск по имени, email, роли.
- Удаление.
- Просмотр одного.

Роли и права.

Возможности супер-админа:

Просмотр/создание/изменение/удаление всех сущностей в системе

Возможности администратора:

1. Добавление/удаление/изменение пользователей
2. Добавление/удаление/изменение категории меню
3. Добавление/удаление/изменение блюд в определенную категорию
4. Просмотр заказов
5. Просмотр отчета по ресторану

+ все возможности официантов

Возможности официантов:

1. Создание/Закрытие заказа
2. Добавление в заказ блюд
3. Удаление блюд из заказа

Работа с категориями меню.

Категория меню
- название (уникальное)
- картинка

Ограничение: в категории может быть много блюд.

Необходимо реализовать CRUD-операции.

- Создание. Название является обязательным полем. Картинка при создании должна сохраняться в хранилище.
- Редактирование. Редактировать можно все поля. При изменении картинки старая картинка должна удалиться из хранилища.
- Просмотр всех. Необходимо реализовать сортировку и поиск по названию.
- Удаление. При удалении сущности картинка также должна удалиться.
- Просмотр одного.

Работа с блюдами.

Блюдо
- название (уникальное)
- картинка
- состав
- калорийность
- цена

Ограничение: блюдо относится только к одной категории меню.

Необходимо реализовать CRUD-операции.

- Создание. Название, калорийность, цена являются обязательными полями. Картинка при создании должна сохраняться в хранилище.
- Редактирование. Редактировать можно все поля. При изменении картинки старая картинка должна удалиться из хранилища.
- Просмотр всех. Необходимо реализовать сортировку и поиск по названию, поиск по составу, сортировку по цене, калорийности.
- Удаление. При удалении сущности картинка также должна удалиться.
- Просмотр одного.

Необходимо реализовать просмотр всех блюд, относящихся к определенной категории.

Необходимо реализовать просмотр меню: всех категорий вместе с относящимися к ним блюдами.

Работа с заказами.

Заказ
- номер (уникальное)

- список блюд
- количество позиций
- итоговая стоимость
- дата создания
- дата закрытия
- официант

Необходимо реализовать CRUD-операции.

- Создание. Все поля обязательны, кроме даты закрытия, числа позиций и итоговой суммы. Число позиций, итоговая стоимость должны быть вычислены при создании (после создания) заказа.
- Редактирование. Редактировать можно все поля, кроме числа позиций и итоговой суммы, а также номера заказа.
- Просмотр всех. Необходимо реализовать сортировку и поиск по номеру, итоговой сумме, дате закрытия, имени официанта.
- Удаление.
- Просмотр одного.