**1. Что такое юзер-сценарии?**

**Юзер-сценарии (User Stories)** — это описание последовательности действий, которые пользователь выполняет для достижения определенной цели в системе. Они помогают понять, как система будет использоваться, и выявить требования.

**Пример юзер-сценариев:**

* **Официант:**
  1. Заходит в систему через планшет.
  2. Выбирает столик из списка.
  3. Добавляет заказ клиента (блюда и напитки).
  4. Отправляет заказ на кухню.
  5. После завершения обслуживания формирует чек и передает его клиенту.
* **Повар:**
  1. Получает уведомление о новом заказе.
  2. Проверяет список блюд, которые нужно приготовить.
  3. Помечает блюдо как готовое после приготовления.
  4. Передает заказ официанту.
* **Менеджер:**
  1. Заходит в систему через личный кабинет.
  2. Проверяет дневной отчет по заказам и продажам.
  3. Управляет меню (добавляет или убирает блюда).
  4. Настраивает доступ для сотрудников.

**2. Что входит в нефункциональные требования?**

**Нефункциональные требования** описывают **качество** системы, но не то, **что она должна делать**. Это требования к производительности, надежности, безопасности, и т.д.

**Пример нефункциональных требований:**

* **Для ресторана:**
  1. **Производительность:**
     + Система должна обрабатывать до 100 заказов в час.
  2. **Надежность:**
     + Доступность системы не менее 99% в течение месяца.
  3. **Безопасность:**
     + Данные сотрудников и клиентов должны быть зашифрованы.
  4. **Юзабилити:**
     + Интерфейс системы должен быть интуитивно понятным, чтобы новые сотрудники могли освоить его за 30 минут.
  5. **Мобильность:**
     + Система должна корректно работать на планшетах с диагональю от 8 дюймов.

**3. Что такое ограничения системы по ТЗ?**

**Ограничения системы** — это факторы, которые ограничивают реализацию или использование системы. Они могут быть связаны с технологиями, временем, бюджетом или законодательством.

**Пример ограничений:**

* **Технологические:**
  + Система должна работать только на планшетах с ОС Android.
  + База данных ограничена на хранение до 10,000 записей.
* **Бюджетные:**
  + Разработка системы не должна превышать 500,000 рублей.
* **Временные:**
  + Система должна быть введена в эксплуатацию через 3 месяца.
* **Законодательные:**
  + Система должна соответствовать требованиям закона о защите персональных данных (например, GDPR или ФЗ-152).

postman

новая коллекция

есть выбор метода запроса

get и post

че они делают - загугли

есть папка с проектом ларавель

app

папочка

открываешь ее

в ней куча других папок

нужна папка route

гуглишь как делать пути в laravel

которые находятся в папке route

там указывается путь с ссылкой и методом запроса

на контроллер

открываешь controllers

или как-то так

папочка такая будет

в какой-то еще папочке скорее всего

там будут перечислены контроллеры

типы

ну

функции

там смотришь

че они делают

логически пытаешься понять код

валидация будет написана

и понимаешь

это либо регистрация

либо выкладывание какого-либо поста

либо я хуй знает

и ты описываешь все это в документе

скорее всего, в постмене

там вроде можно было описать

че принимает

какие параметры

запрос

принимает

че да как

вот примерно так

**Как все это сделать**

### ****Шаблон для работы с Laravel приложением:****

#### ****1. Изучение маршрутов (Routes)****

* **Где искать:**  
  Открой файл маршрутов приложения в папке routes. Обычно это:
  + Для веб-приложения: routes/web.php
  + Для API: routes/api.php
* **Что делать:**
  + Найди все маршруты в файле.
  + Изучи их структуру: что они делают, какие контроллеры и методы вызываются.
  + Помести информацию в таблицу или документ, где будет указано:
    - Путь маршрута (URL).
    - Тип запроса (GET, POST и т.д.).
    - Контроллер и метод, которые обрабатывают маршрут.

**Шаблон для записи маршрутов:**

Путь маршрута: /users

Тип запроса: GET

Контроллер: UserController

Метод: index

Путь маршрута: /users/store

Тип запроса: POST

Контроллер: UserController

Метод: store

#### ****2. Изучение контроллеров (Controllers)****

* **Где искать:**  
  Контроллеры обычно находятся в папке app/Http/Controllers.
* **Что делать:**
  1. Открой каждый контроллер и посмотри, какие методы в нем реализованы.
  2. Соотнеси методы с маршрутом, который они обрабатывают.
  3. Определи, что именно делает каждый метод (например, получает данные из базы, отображает представление, обрабатывает запрос и т.д.).

**Шаблон для записи контроллеров:**

Контроллер: UserController

Методы:

- index() — выводит список пользователей.

- store(Request $request) — сохраняет нового пользователя.

#### ****3. Тестирование запросов в Postman****

* **Где искать:**  
  Для тестирования запросов открой Postman и создай коллекцию запросов для каждого маршрута.
* **Что делать:**
  1. Создай новую коллекцию в Postman.
  2. Добавь запросы по каждому маршруту:
     + Для GET-запросов укажи URL маршрута и параметров (если есть).
     + Для POST-запросов укажи URL и тело запроса (формат JSON или другие параметры).
  3. Укажи ожидаемый ответ от сервера (например, JSON-объект, статус 200 и т.д.).

**Шаблон для записи запросов в Postman:**

Запрос:

Метод: GET

URL: http://yourdomain.com/users

Ожидаемый ответ:

- Статус: 200 OK

- Данные: Список пользователей в формате JSON

Запрос:

Метод: POST

URL: http://yourdomain.com/users/store

Тело запроса:

{

"name": "John Doe",

"email": "johndoe@example.com"

}

Ожидаемый ответ:

- Статус: 201 Created

- Данные: JSON с информацией о созданном пользователе.

#### ****4. Документация запросов в Postman****

* **Где искать:**  
  После тестирования запросов в Postman, создайте коллекцию и задокументируйте ее.
* **Что делать:**
  1. В Postman создай коллекцию для всех запросов.
  2. Добавь описание для каждого запроса: что он делает, какие параметры принимает, какой результат возвращает.
  3. Экспортируй коллекцию в формате JSON или в другом удобном виде.

**Шаблон для документации в Postman:**

Название коллекции: Запросы API для приложения

Запрос 1: Получение списка пользователей

- Метод: GET

- URL: /users

- Описание: Этот запрос возвращает список всех пользователей.

- Параметры: Нет.

- Ожидаемый ответ:

{

"data": [

{"id": 1, "name": "John Doe", "email": "johndoe@example.com"},

{"id": 2, "name": "Jane Doe", "email": "janedoe@example.com"}

]

}

Запрос 2: Создание пользователя

- Метод: POST

- URL: /users/store

- Описание: Этот запрос создает нового пользователя.

- Параметры:

- name (string) — Имя пользователя.

- email (string) — Электронная почта пользователя.

- Ожидаемый ответ:

{

"id": 3,

"name": "New User",

"email": "newuser@example.com"

}

#### ****5. Написание технического задания (ТЗ)****

* **Где искать:**  
  Для ТЗ возьми всю информацию, которую ты собрал по маршрутам, контроллерам и запросам.
* **Что делать:**
  1. Создай документ, в котором перечисляешь все маршруты, контроллеры и методы.
  2. Укажи описание каждого маршрута, его назначение и возможные параметры.
  3. Опиши логику работы приложения, основываясь на контроллерах и запросах.
  4. Укажи ожидаемые ответы для каждого запроса (статусы, данные).

**Шаблон для ТЗ:**

1. \*\*Маршрут /users\*\*

- Тип запроса: GET

- Контроллер: UserController

- Метод: index

- Описание: Возвращает список пользователей.

- Ожидаемый ответ:

- Статус: 200 OK

- Данные: JSON с пользователями.

2. \*\*Маршрут /users/store\*\*

- Тип запроса: POST

- Контроллер: UserController

- Метод: store

- Описание: Создает нового пользователя.

- Параметры:

- name: имя пользователя

- email: электронная почта пользователя

- Ожидаемый ответ:

- Статус: 201 Created

- Данные: JSON с информацией о новом пользователе.

### ****Резюме****

В результате выполнения задания ты получишь:

1. Подробный список маршрутов с типами запросов и контроллерами.
2. Описание контроллеров и их методов.
3. Тестирование всех маршрутов через Postman с документацией.
4. Написание ТЗ для дальнейшей разработки и документации системы.