Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа № 7

**Проектирование веб-API**

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

Проверил: кандидат технических наук

Ржавин Вячеслав Валентинович

Чебоксары, 2025

Таблица целей API

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Пользователи | Действия | Этапы | Входные данные | Выходные данные | Цели |
|  | *Кто пользователи?* | *Что они могут сделать?* | *Как они это делают?* | *Что им нужно? Откуда это?* | *Что они получают? Как это используется* | *Переформулируйте*  *действия+входные*  *данные+результаты* |
| 1 | Арендатор | Арендовать участок | Находят участок | Каталог  Запрос | Участки, походящие под запрос | Найти участки в каталоге при помощи запроса |
| Смотрят графики бронирования | Запрос  Выбранный участок | Календарные даты, свободные, занятые | Просмотреть графики броней, выбрать удобный |
| Подают заявку на бронирование | Выбранный участок  Данные арендатора | Заявка | Подать заявку на аренду |
| Просмотреть историю бронирований | Смотрят историю | Запрос | Список всех бронирований арендатора | Просмотреть историю всех броней |
| Управлять своим профилем | Изменяют свои данные | Новые данные от пользователя | Обновленные данные в бд | Изменить данные арендатора |
| 2 | Арендодатель | Управлять участками | Добавляют участок | Данные участка | Новый участок в каталоге | Добавить новый участок в бд для последующей сдачи |
| Изменяют данные участка | Новые данные участка | Обновленные данные в бд | Изменить данные участка |
| Удаляют участок | Номер участка | Удаление участка | Удалить участок |
| Управляют заявками | Просматривают заявки | Запрос к бд | Заявки по своим участкам | Просмотреть заявки на аренду своих участков |
| Одобряют заявки | Номер заявки | Обновление статуса заявки, создание шаблона договора | Одобрить заявку и начать подготовку договора |
| Отклоняют заявки | Номер заявки | Обновление статуса заявки | Отклонить заявку |
| 3 | Председатель | Просматривает отчеты по аренде | Формирует отчет по сданным участкам | Параметры отчета | Отчет | Получить общий взгляд на сдачу участков |
|  |  |  | Формирует финансовый отчет | Параметры отчета | Отчет | Проанализировать финансы кооператива |

Устранение влияния поставщика при проектировании API

|  |  |
| --- | --- |
| Влияние | Результаты изменения |
| Данных | Отойти от модели данных |
| Кода и бизнес-логики | Инкапсулирование логики |
| Архитектуры и ПО | Скрытие внутренних сервисов |
| Организации | Скрытие внутренних отделов |

Соотнесите полученную таблицу целей API с полученной в предыдущей лабораторной работе диаграммой вариантов использования.

|  |  |  |
| --- | --- | --- |
| Субъект | Целевой вариант использования | Расширения и включения |
| Арендатор | UC 1. Просмотр доступных участков | UC1.1. просмотреть участок  UC1.2. просмотреть бронирования |
|  | UC 2. Подача заявки на аренду | - |
|  | UC 3. Оплата аренды | - |
|  | UC 4. Управление профилем | UC 4.1. изменение данных  UC 4.2. просмотр истории бронирований |
|  | UC 5. Заключение договора | UC 5.1. подписание акта приема-передачи  UC 5.2. проведение инструктажа  UC 5.3. передача ключей |
| Арендодатель | UC 6. Управление участками | UC 5.1. добавление нового участка  UC 5.2. редактирование данных участков  UC 5.3. удаление участка |
|  | UC 7. Управление заявками | UC 6.1. просмотр заявок  UC 6.2. одобрение заявки  UC 6.3. отклонение заявки |
|  | UC 8. Управление профилем | UC 8.1. изменение данных  UC 8.2. просмотр участков |
| Председатель | UC 9. Просмотр отчетов | - |

Маршруты

|  |  |  |  |
| --- | --- | --- | --- |
| Субъект | Use case | Маршрут | HTTP-метод |
| Арендатор | UC 1 | /plots | Get |
|  | UC 1.1 | /plots/{id} | Get |
|  | UC 1.2 | /plots/{id}/shedule | Get |
|  | UC 2 | /applications | Post |
|  | UC 3 | /applications/{id}/payment | Get |
|  | UC 4 | /profile | Get |
|  | UC 4.1 | /profile/info | Put |
|  | UC 4.2 | /profile/shedules | Get |
|  | UC 5 | /applications/{id}/contract | Post |
|  | UC 5.1 | /applications/{id}/acceptance\_act | Pos |
|  | UC 5.2 | /applications/{id}/briefing | Post |
|  | UC 5.3 | /applications/{id}keys\_receive | Post |
|  | UC 6 | /plots | Get |
|  | UC 6.1 | /plots | Post |
|  | UC 6.2 | /plots | Put |
|  | UC 6.3 | /plots/{id} | Delete |
|  | UC 7 | /applications | Get |
|  | UC 7.1 | /applications | Get |
|  | UC 7.2 | /applications/{id}/approve | Put |
|  | UC 7.3 | /applications/{id}/reject | Put |
|  | UC 8 | /profile | Get |
|  | UC 8.1 | /profile/info | Put |
|  | UC 8.2 | /profile/plots/mine | Get |
|  | UC 9 | /reports | Get |

Структура данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ресурс | Поле | Описание свойства | Тип данных | Обязательно |
| Участок | Id | Идентификатор участка | integer | да |
| Owner\_id | FK на арендодателя | integer | да |
| Renter\_id | FK на арендатора | integer | нет |
| Price\_item\_id | FK на строку прейскуранта | integer | да |
| Address | Адрес участка | string | да |
| Area | Площадь участка | number | да |
| Description | Описание участка | string | да |
| Арендатор | Id | ID арендатора | integer | да |
| Name | Фио арендатора | string | да |
| Phone | Номер арендатора | string | да |
| Email | Почта арендатора | string | нет |
| Арендодатель | Id | ID арендодателя | integer | да |
| Name | Фио арендодателя | string | да |
| Phone | Номер арендодателя | string | да |
| Email | Почта арендодателя | string | нет |
| Договор аренды | Id | ID договора | integer | да |
| Plot\_id | FK на участок | integer | да |
| Renter\_id | FK на арендатора | integer | да |
| Start\_date | Дата начала аренды | date | да |
| End\_date | Дата конца аренды | date | да |
| Status | Статус договора | string | да |
| Прейскурант | Id | ID прейскуранта | integer | да |
| Name | Имя прейскуранта | string | да |
| Valid\_from | Дата начала действия прейскуранта | date | да |
| Valid\_to | Дата конца действия прейскуранта | date | да |
| Строка прейскуранта | Id | ID строки | integer | да |
| Price\_list\_id | FK на прейскурант | integer | да |
| Price\_per\_day | Стоимость аренды за день | number | да |
| Оплата | Id | ID оплаты | integer | да |
| Contract\_id | FK на договор | integer | да |
| Amount | Сумма оплаты | number | да |
| Payment\_date | Дата оплаты | date | да |
| status | Статус оплаты | string | да |

Ответы на вопросы

1. Что еще, кроме REST API, можно использовать для организации связи между приложениями?

Программные комплексы, разработанные в соответствии с СОА, обычно реализуются как набор веб-служб, взаимодействующих по протоколу SOAP, но существуют и другие реализации (например, на базе jini, CORBA, на основе REST).

1. Какие типы API существуют?

Приватны, публичные, партнерские

1. Приведите примеры использования API

Когда вы создаёте новую учётную запись в приложении, вам часто предлагается зарегистрироваться с помощью аккаунта Facebook или Google. Вы когда-нибудь задумывались, как это работает?

Такие приложения, как Facebook и Google, уже имеют доступ к вашим данным. Вместо того чтобы выполнять утомительную, дорогостоящую и рискованную задачу по сбору информации о каждом пользователе, приложение обращается к базам данных Google и Facebook через API для проверки ваших учётных данных.

Преимущества этого API для приложения заключаются в меньшем риске, более простом внедрении и более быстром процессе превращения вас в пользователя. Преимущества для Facebook и Google заключаются в расширении охвата и получении дополнительной информации о том, как и к чему вы обращаетесь, которую они могут предоставить рекламодателям.

1. Что такое спецификация API?

Это документ, описывающий как взаимодействовать с апи.

1. Что такое конечная точка API?

Это url адрес по которому по которому приложение может обратиться к апи для выполнения определенной операции.

Конечная точка API — это место, где выполняются эти запросы (известные как вызовы API)

1. Какие задачи решаются на этапе предварительного проектирования API

Определение Целей и Бизнес-Требований

Проектирование Модели Данных и Ресурсов

Определение Эндпоинтов и Операций

Определение Стратегии Аутентификации и Авторизации

1. Какие команды HTTP являются идемпотентными

Метод HTTP является идемпотентным, если повторный идентичный запрос, сделанный один или несколько раз подряд, имеет один и тот же эффект, не изменяющий состояние сервера. Другими словами, идемпотентный метод не должен иметь никаких побочных эффектов (side-effects), кроме сбора статистики или подобных операций. Корректно реализованные методы [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Reference/Methods/GET), [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Reference/Methods/HEAD), [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Reference/Methods/PUT) и [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Reference/Methods/DELETE) **идемпотентны**, но не метод [POST](https://developer.mozilla.org/ru/docs/Web/HTTP/Reference/Methods/POST). Также все [безопасные](https://developer.mozilla.org/ru/docs/Glossary/Safe) методы являются идемпотентными.

1. Существует ли клиентский веб API?

[Клиентский](https://en.wikipedia.org/wiki/Client-side) веб-API — это программный интерфейс для расширения функциональности [веб-браузера](https://en.wikipedia.org/wiki/Web_browser) или другого HTTP-клиента. Изначально они чаще всего представляли собой нативные [плагины](https://en.wikipedia.org/wiki/Plug-in_(computing)) [расширения браузера](https://en.wikipedia.org/wiki/Browser_extension), однако большинство новых API ориентированы на стандартизированные [привязки JavaScript](https://en.wikipedia.org/wiki/JavaScript).

1. Из чего состоит серверный веб-API ?

[Серверная](https://en.wikipedia.org/wiki/Server-side) веб-API состоит из одной или нескольких общедоступных *конечных точек* с определённой системой [запросов и ответов](https://en.wikipedia.org/wiki/Request%E2%80%93response), обычно представленной в формате [JSON](https://en.wikipedia.org/wiki/JSON) или [XML](https://en.wikipedia.org/wiki/XML). Веб-API чаще всего предоставляется с помощью [веб-сервера](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) на базе [HTTP](https://en.wikipedia.org/wiki/Web_server).

1. Как в REST API передаются данные на сервер?

В REST API данные передаются на сервер через несколько основных механизмов. Два самых распространенных — это **строка запроса (Query String)** и **тело запроса (Request Body)**. Выбор способа зависит от типа операции и передаваемых данных.

**Строка запроса (Query String)**

**Что это:** Часть URL, которая идет после знака ? и содержит пары "ключ=значение", разделенные амперсандом &.

<https://api.example.com/users?param1=value1&param2=value2&param3=value3>

**Тело запроса (Request Body)**

**Что это:** Данные, которые передаются внутри HTTP-запроса. Для их описания обычно используются заголовки Content-Type и Content-Length.

**Синтаксис:** Зависит от формата, указанного в Content-Type.

**Когда используется:**

* Для операций, которые **изменяют состояние сервера**: создание (POST), обновление (PUT, PATCH).
* Для передачи **больших объемов данных** или **сложных структур** (вложенные объекты, массивы).

1. Из каких шагов складывается обработка API-запросов? (маршрутизация запроса в соответствующую службу, проверка данных, логика, …)

Обработка API-запроса — это цепочка последовательных шагов через которые проходит запрос, прежде чем будет сформирован ответ.

**Маршрутизация (Routing)**

* **Что происходит:** Сервер определяет, **какой конкретно код (контроллер, функция, обработчик)** должен выполнить этот запрос.
* **Как работает:** Сопоставляет **HTTP-метод** (GET, POST и т.д.) и **URL-путь** (/api/users/123) с заранее объявленным маршрутом в приложении.

**Парсинг и валидация данных (Parsing & Validation)**

* **Парсинг:** Сервер извлекает данные из запроса и преобразует их в удобный для программы формат.
  + **Query String** парсится в объект ключ-значение.
  + **Тело запроса (Body)** парсится из JSON, form-data и т.д. в объект или модель.
  + **Параметры пути (Route Parameters)** извлекаются из URL (например, id=123 из /users/123).
* **Валидация:** Проверка, что переданные данные корректны и соответствуют ожиданиям.
  + *Примеры проверок:* обязательные поля присутствуют, email имеет правильный формат, число находится в допустимом диапазоне, тип данных верный.

**Работа с данными (Data Access / Persistence)**

* **Что происходит:** На этом шаге бизнес-логика взаимодействует с уровнем данных.
* **Как работает:** Выполняются запросы к базе данных (SQL, NoSQL), чтение/запись в кэш (Redis), обращение к внешним API или файловой системе.
* **Цель:** Сохранить или извлечь данные, необходимые для выполнения бизнес-логики.

**Формирование ответа (Response Forming)**

* **Что происходит:** Сервер подготавливает данные, которые будут отправлены обратно клиенту.
* **Как работает:**
  + Выбирается **HTTP-статус** (200 OK, 201 Created, 404 Not Found).
  + Данные сериализуются в нужный формат (обычно JSON).
  + Добавляются необходимые **заголовки (Headers)**.
* **Пример:** После создания заказа возвращается статус 201 Created, в заголовке Location указывается ссылка на новый ресурс, а в теле — JSON с данными заказа.

1. Продолжите предложение «При разработке API важно иметь глубокие и точные знания о том, кто может использовать API; …