**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**на разработку распределенной системы обработки информации**

**«Автоматизированная система контроля качества городской среды»**

**По дисциплине РСОИ**

**Москва 2022 г.**

**Оглавление**

[1. Аналитический раздел 3](#_Toc104043993)

[1.1. Краткое описание предметной области 3](#_Toc104043994)

[1.2. Существующие аналоги 3](#_Toc104043995)

[1.3. Описание системы 3](#_Toc104043996)

[1.4. Назначение разработки 3](#_Toc104043997)

[1.5. Требования к системе 4](#_Toc104043998)

[1.6. Требования к функциональным характеристикам 4](#_Toc104043999)

[1.7. Функциональные требования к порталу с точки зрения пользователя 5](#_Toc104044000)

[1.8. Входные параметры системы 6](#_Toc104044001)

[1.9. Выходные параметры системы 7](#_Toc104044002)

[1.10. Требования к составу и параметрам технических средств 7](#_Toc104044003)

[1.11. Требования к надежности 8](#_Toc104044004)

[2. Конструкторский раздел 9](#_Toc104044005)

[2.1. Проектирование серверного ПО на основе микросервисной архитектуры 9](#_Toc104044006)

[2.2. Топология системы 10](#_Toc104044007)

[2.3. Описание используемых алгоритмов 11](#_Toc104044008)

[2.4. Диаграмма вариантов использования (Use-Case) 12](#_Toc104044009)

[2.5. Диаграмма последовательности (Sequence) 15](#_Toc104044010)

[2.6. Диаграмма потоков данных 16](#_Toc104044011)

[3. Технологический раздел 17](#_Toc104044012)

[3.1. Описание REST API 17](#_Toc104044013)

[3.2. Структура базы данных 18](#_Toc104044014)

[3.3. Тестирование 20](#_Toc104044015)

[Заключение 22](#_Toc104044016)

1. **Аналитический раздел**
   1. **Краткое описание предметной области**

В настоящее время развитие городской среды, а точнее городской инфраструктуры является наиболее актуальной проблемой в связи с очень активным заселением как крупных, так и малых городов. Состояние городской среды практически любого не крупного российского города можно охарактеризовать как кризисное. Формирование развитой инфраструктуры является очень важным фактором обеспечения достойного уровня условий жизни местного населения. Развитие благоприятной среды обитания достигается путем вложения огромных денежных средств в реконструкцию и создание новых зон отдыха, благоустройство дворов, обновление дорожно- транспортной инфраструктуры, в том числе остановочных комплексов, но доступного механизма контроля уже реализованных проектов для средних и малых городов не существует, а если и существует, то для очень небольшого количества.

* 1. **Существующие аналоги**

Среди аналогов можно отметить порталы ag.mos.ru и narod-expert.ru. Данный проект должен иметь следующие преимущества перед существующими аналогами:

* Возможность запуска системы не только в больших и средних городах, но и малых городах, которые не могут позволить разработку данного портала.
* Частично автоматизированный поиск актуальной информации о существующих нарушениях городской среды.
* Удобный интерфейс и высокая скорость загрузки страниц портала.
  1. **Описание системы**

Проект должен представлять собой портал для сбора информации о существующих недостатках городской среды как путём ручного ввода реальных пользователей системы, так и автоматизированным сбором в режиме реального время из открытых источников, которые доступны в сети интернет. На основе собранной информации система сможет сформировать общую статистику по недостаткам и предоставить её пользователям системы. В данной информационной системе предполагается наличие трёх типов пользователей: пользователь, модератор, ответственный представитель от администрации.

* 1. **Назначение разработки**

Главное назначение разрабатываемого портала – возможность выявления недостатков городской среды. Портал должен работать выполнять полностью свой функционал не зависимо от размеров и географического расположения любого города РФ, а также способствовать развитию городской инфраструктуры и окружающей среды.

* 1. **Требования к системе**

1. Время восстановления системы после сбоя не должно превышать 15 минут.

Для проверки данного требования будет специально выключен один из микросервисов путём вызова команды в соответствующем контейнере команды “sudo kill”, после чего механизмы docker’а должны автоматически перезапустить данный контейнер.

1. Каждый узел должен автоматически восстанавливаться после сбоя.

Для проверки данного требования будет выводиться строго только один микросервис, и он обязательно должен будет иметь способность восстанавливаться, как это описано выше.

* 1. **Требования к функциональным характеристикам**

1. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы пользователя на получение информации не должна превышать 23 секунды без учета латентности географического расположения узла.

Проверяться это требованием будет путём сверки времени загрузки страницы в инспекторе кода как это показано на изображении ниже.

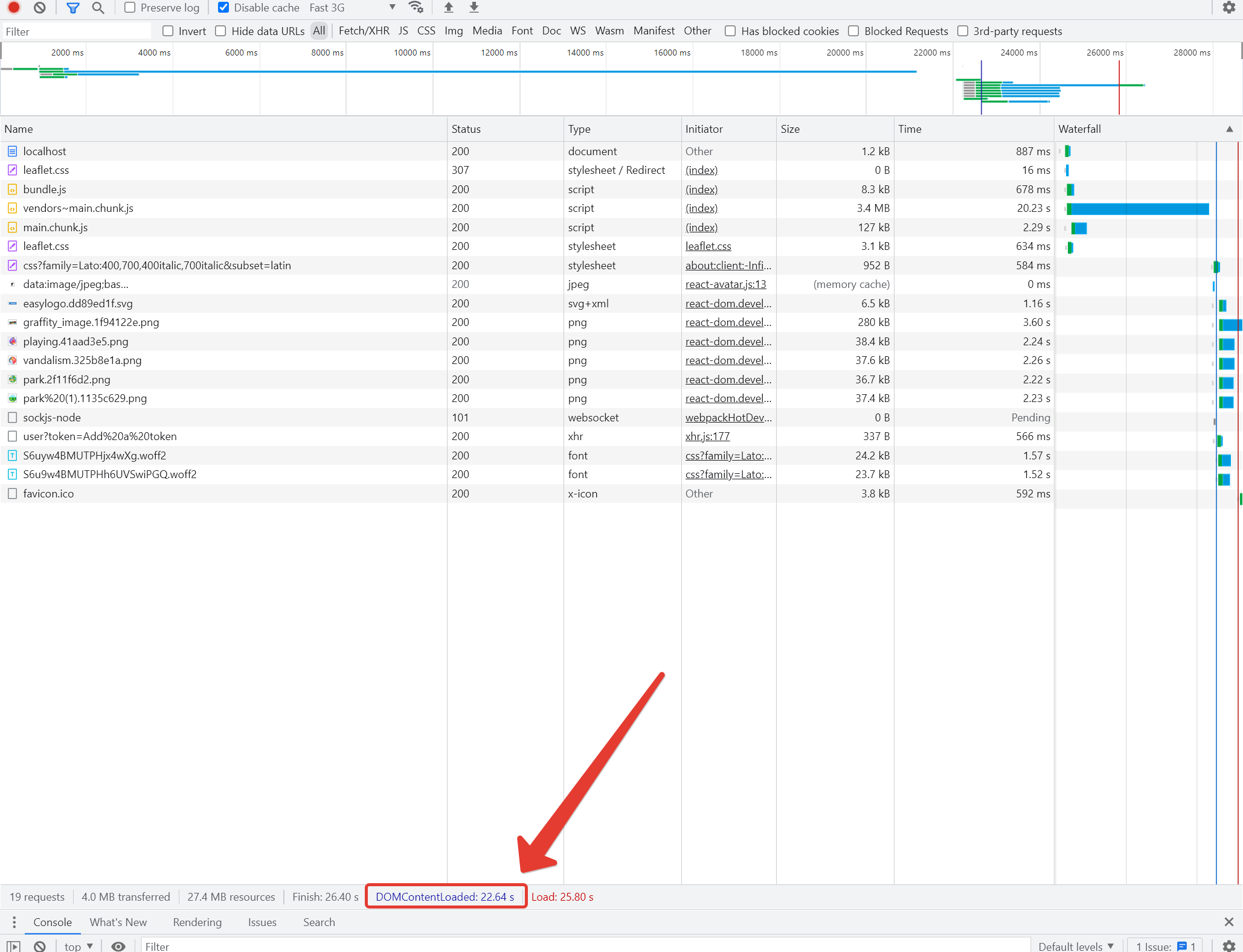


Рисунок 1.1. Скриншот инспектора кода в браузере Google Chrome.

1. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы, добавляющие или изменяющие информацию на портале, не должна превышать 5 секунд без учета латентности географического расположения узла.

Проверяться это требованием будет путём сверки времени загрузки страницы в инспекторе кода.

1. Портал должен обеспечивать возможность запуска в современных браузерах: не менее 85% пользователей Интернета должны иметь возможность пользоваться порталом без какой-либо деградации функционала.

Проверяться этот показатель будет путём запуска сайта и проверки его соответствия эталону (который будет определён в ходе разработки) в следующих браузерах: Google Chrome 100.0.4896.69 (64 бит), Microsoft Edge 100.0.1185.27 beta (64-разрядная версия), Opera 85.0.4341.60.

1. Система должна иметь систему авторизации, для обеспечения безопасности и разграничения прав доступа.

Проверяться данное требование будет путём попытки авторизации зная логин пользователя, но не зная пароль. Или попытка закрыть дело о недостатке с правами обычного пользователя (не представителя администрации или модератора).

* 1. **Функциональные требования к порталу с точки зрения пользователя**

Портал должен обеспечивать реализацию следующих функций:

1. Система должна обеспечивать регистрацию как представителю администрации, обычному пользователю, так и модератору. (Примечание, при регистрации у зарегистрированного человека, будут по умолчанию права обычного пользователя, роль в будущем ему может сменить модератор, на “Представитель администрации”, “Модератор” или “Обычный пользователь”).
2. Система должна обеспечивать авторизацию пользователей.
3. Система должна обеспечивать разделение пользователей на 3 роли:

* представитель администрации;
* пользователь;
* модератор.

1. Система должна предоставлять **представителю администрации** следующие функции:

* изменение информации о зафиксированных недостатках (например, сменить статус с “ожидание” в “рассматривается” или “рассматривается” в “ведутся работы”);
* добавление информации о проходящих работах в виде комментария под обращением;
* выгрузку обращений в формате pdf;
* возможность ответить на обращение;
* изменение личной информации в аккаунте;

1. Система должна предоставлять **пользователю** следующие функции:

* просмотр информации о зафиксированных нарушениях и недостатках города, другими пользователями: тип нарушения, описание, местоположение, фотографии, время события;
* добавление и изменение зафиксированных нарушений;
* изменение личной информации в аккаунте;

1. Система должна предоставлять **модератору** следующие функции:

* неограниченные полномочия по изменению контента на портале;
* подтверждение зафиксированных нарушений, после чего будет происходить автоматическая отправка их ответственному представителю по электронной почте в формате pdf;
* назначение роли любому зарегистрированному пользователю, например роли “модератора” или “представителя администрации”;
* изменение личной информации в аккаунте.
  1. **Входные параметры системы**

**Официальный представитель администрации**

* Наименование юридического лица, максимальная длина – 512 символов.
* Адрес электронный почты представителя – 256 символов.
* Имя, максимальная длина – 256 символов.
* Фамилия, максимальная длина – 256 символов.
* Отчество, максимальная длина – 256 символов.
* Географическое положение организации. Максимальная длина текста – 512 символов. Например, «г. Москва, ул. Пушкина, дом 76»

**Пользователь**

* Ник (Псевдоним), максимальная длина – 256 символов.
* Имя, максимальная длина – 256 символов.
* Фамилия, максимальная длина – 256 символов.
* Отчество, максимальная длина – 256 символов.
* Дата рождения, в формате «День.Месяц.Год».
* Место проживания (допускается приблизительное). Максимальная длина текста – 512 символов. Например, «г. Москва, ул. Пушкина, дом 76»
* Адрес электронный почты – 256 символов.

**Модератор**

* Ник (Псевдоним), максимальная длина – 256 символов.
* Имя, максимальная длина – 256 символов.
* Фамилия, максимальная длина – 256 символов.
* Отчество, максимальная длина – 256 символов.

**Социальная сеть ВК:**

* Массив постов, с отмеченным геоточками в пределах города (области, района) – 1000 постов.
  1. **Выходные параметры системы**

Выходными параметрами системы являются веб-страницы. Они должны содержать

следующую информацию:

* список возможных недостатков;
* детальная информация о выбранном недостатке (название, фото, описание, адрес, тип нарушения, время фиксации, местоположение);
* общая статистика по округам города, в срезе нарушений зафиксированными самостоятельно пользователями и автоматически зафиксированных;
* информация об аккаунте пользователя;
  1. **Требования к составу и параметрам технических средств**

Все серверные приложения должны потреблять суммарно не более 16 Гб оперативной памяти и работать на сервере с процессором Intel i7 2.4 GHz и жестким диском SATA 100 IOPS 100 Гб.

* 1. **Требования к надежности**

Система должна работать в соответствии с данным техническим заданием без рестарта. Для проверки данного требования, система будет запущена с умеренной нагрузкой (2-3 одновременно работающих пользователя) и ни один из микросервисов не должен аварийно остановится.

1. **Конструкторский раздел**

Каждая информационная система должна обеспечивать требуемую производительность, функциональность, безопасность, безотказную работу, пропускную способность и множество других важнейших для эффективной работы факторов. Это достигается путем грамотного проектирования системы.

В данном разделе курсового проекта будет описана архитектура и алгоритмы разрабатываемой системы, описаны отдельные компоненты системы, а также спроектированы основные диаграммы, описывающие работу системы и взаимодействие отдельных компонентов.

* 1. **Проектирование серверного ПО на основе микросервисной архитектуры**

Серверная часть разрабатываемого приложения представляет собой совокупность микросервисов — процессов, запущенных в операционной системе. Каждый микросервис выполняет определенный для него функционал, реализуя основную логику работы приложения по хранению данных и контролю доступа к ним. Важно отметить, что вся бизнес-логика обработки данных выполняется на клиентской стороне, что позволяет снизить нагрузку на сервера и применять, в дальнейшем, разрабатываемое серверное программное обеспечение в различных предметных областях без необходимости вносить какие-либо изменения.

Для хранения различных данных, требуемых для корректной обработки клиентских запросов, на серверной машине развернута СУБД PostgreSQL, MINIO и Redis.

Основной сценарий, отражающий процесс обработки клиентского запроса, выглядит следующим образом: микросервис разбирает запрос, выделяя из него команду, необходимую для выполнения, и данные, над которыми будет выполнена та или иная операция. После этого микросервис приступает к обработке запроса, в ходе которой он может взаимодействовать с сервером баз данных или другими микросервисами, запрашивая у них дополнительные данные. По окончании обработки микросервис возвращает клиенту ответ, в теле которого содержатся данные, определенные протоколом взаимодействия между клиентской и серверной частями приложения.

В существующей архитектуре выделяется четыре микросервиса: reactfrontend, apigateway, appeals, users, mailservice. Каждый микросервис связан с набором сущностей, информацию о которых он способен обрабатывать. Сущности лежат в основе архитектуры базы данных, описанной далее.

* 1. **Топология системы**

Система будет состоять из фронтэнда и пяти сервисов, что наиболее целесообразно для реализации ее основного назначения. Топология разрабатываемой системы представлена на рисунке 2.1.



Рисунок 2.1 – Топология системы

**Файловый сервис** отвечает за хранение бинарных файлов, которые могут быть загружены пользователем, он реализует следующие функции:

* загрузка бинарного файла;
* показ/скачивание конкретного бинарного файла (по id);

**Сервис обращений** отвечает за хранение информации о представленных на портале обращениях и реализует следующие функции:

* получение списка обращений;
* получение информации о конкретном обращении;
* добавление нового обращения;
* изменение информации конкретного обращения;
* удаление обращения;

**Сервис пользователей** отвечает за пользователей портала и реализует следующие функции:

* регистрацию пользователя;
* получение данных пользователя;
* удаление пользователя
* изменение данных пользователя

**Сервис электронной почты** отвечает за отправку писем на произвольные email адреса и реализует следующие функции:

* отправка писем на электронную почту.

**Сервис координатор (API Gateway)** отвечает за диспетчеризацию запросов и предоставляет оптимальный унифицированный API.

**Фронтэнд** принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа фронтэнд выполняет запросы к координационному сервису, получает ответы и отсылает их пользователю.

* 1. **Описание используемых алгоритмов**

Все сервисы разрабатываемого портала используют протокол HTTP для получения и отправления информации. Все запросы, кроме получения токена проходят через координационный сервис.

Приложение использует JWT для проверки аутентификации пользователя следующим образом:

* + - 1. Сперва пользователь заходит на сервер аутентификации с помощью аутентификационного ключа (это может быть пара логин/пароль).
      2. Затем сервер аутентификации создает JWT и отправляет его пользователю.
      3. Когда пользователь делает запрос к API приложения, он добавляет к нему полученный ранее JWT.
      4. Когда пользователь делает API запрос, приложение может проверить по переданному с запросом JWT является ли пользователь тем, за кого себя выдает. В этой схеме сервер приложения сконфигурирован так, что сможет проверить, является ли входящий JWT именно тем, что был создан сервером аутентификации (процесс проверки будет объяснен позже более детально).

Типичный алгоритм аутентификации на основе токенов представлен на рисунке 2.2.

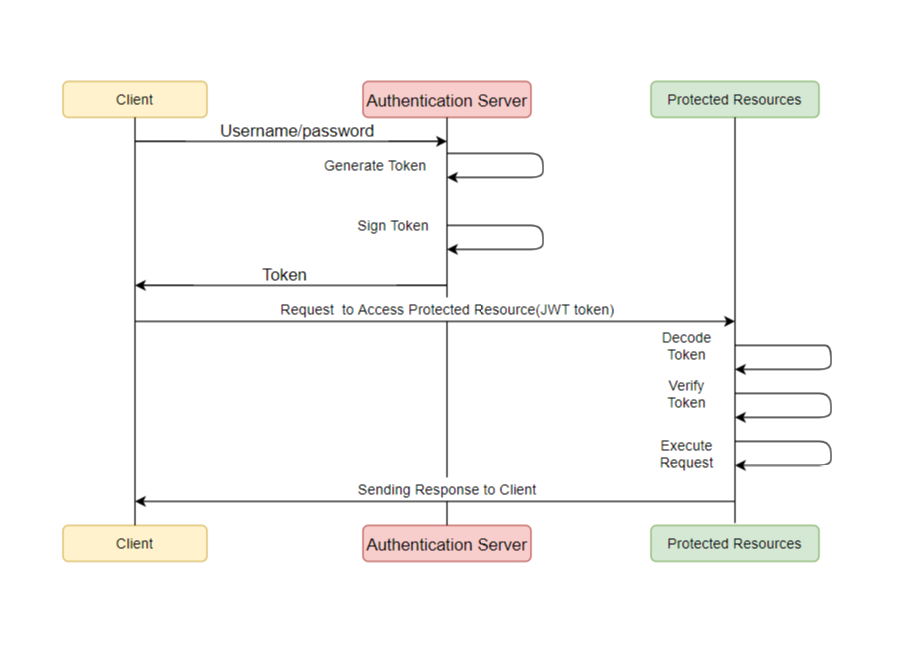


Рисунок 2.2 – Аутентификация на основе токенов

Помимо этого, из важных особенностей реализации стоит отметить, что каждый сервис, кроме сервиса координатора имеет свое собственное хранилище данных, доступ к которому есть только у него.

Также, в системе организована пагинация некоторых запросов для уменьшения поискового трафика и фильтрация данных, выполняемая на фронтэнде.

* 1. Диаграмма вариантов использования (Use-Case)

Диаграммы вариантов использования (use case diagram), описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы вариантов использования преследует цели:

* Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
* Сформулировать общие требования к функциональному поведению проектируемой системы.
* Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
* Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой. Рассматривая диаграмму вариантов использования в качестве модели системы, можно ассоциировать ее с моделью черного ящика". Действительно, подробная детализация данной диаграммы на начальном этапе проектирования скорее имеет отрицательный характер, поскольку предопределяет способы реализации поведения системы. А согласно рекомендациям именно эти аспекты должны быть скрыты от разработчика на диаграмме вариантов использования. В самом общем случае, диаграмма вариантов использования представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров, возможно, некоторых интерфейсов, и отношений между этими элементами. При этом отдельные компоненты диаграммы могут быть заключены в прямоугольник, который обозначает проектируемую систему в целом. Следует отметить, что отношениями данного графа могут быть только некоторые фиксированные типы взаимосвязей между актерами и вариантами использования, которые в совокупности описывают сервисы или функциональные требования к моделируемой системе.

Разработанная диаграмма вариантов использования для актеров пользователь, представитель администрации и модератор представлена на рисунке 2.3 соответственно.



Рисунок 2.3 – Use Case диаграмма разрабатываемой платформы

* 1. Диаграмма последовательности (Sequence)

На диаграммах последовательностей, иногда называемых сценариями, показываются объекты и сообщения, которыми они обмениваются. Каждый объект изображается в виде вертикальной линии («линии жизни» объекта). По вертикали сверху вниз направлена временная ось. Сообщение, показываемое в виде стрелки от объекта к объекту, соответствует вызову операции соответствующего класса. Таким образом, на диаграмме можно показать поток сообщений во времени (сценарий). С помощью диаграмм этого вида можно описать как основной, так и альтернативные потоки событий.



Рисунок 2.4. Диаграмма последовательности действий при запросе пользователем

Главный сервис приложения отправляет запрос на получение обращений, которые доступны пользователю. После получения данных главный сервис обращается к сервису обращений, чтобы для каждого автора или категории получить список обращений, которые были опубликованы. После окончания данной процедуры главный сервис ранжирует обращения по времени публикации, формирует веб-страницу и возвращает ее пользователю.

* 1. Диаграмма потоков данных

Рассматриваемая система предполагает распределенное хранение данных. Все данные системы предполагают хранение в единой базе данных, хранилищами данных являются таблицы. Диаграмма потоков данных, представленная на рисунке 2.5., отображает модель информационной системы с точки зрения хранения, передачи и обработки данных во время обработки запроса пользователя на получение новых обращений.



Рисунок 2.5. Диаграмма потоков данных разрабатываемой системы на примере получения информации об обращениях

1. Технологический раздел

На данном этапе разработки информационного портала будет произведено описание типов и структур данных в нотации IDEF1x, а также тестирование, обработка ошибок и поведение системы в случае отказа.

* 1. **Описание REST API**

Разработка API один из важнейших этапов создания микросервиса, так как от полноты реализации функционала API зависит удобство реализации остальных микросервисов и производительность мессенджера. API, описанное в данной курсовой работе, соответствует архитектурному стилю REST и поддерживает следующие функции:

* Создание пользователя (UsersService);
* Обновление данных пользователя (смена логина, ФИО, пароля и т.д.) (UsersService);
* Удаление пользователя (UsersService);
* Получение пользователя по ID (UsersService);
* Получение всех/подвыборки пользователей (UsersService);
* Получение всех/подвыборки обращений (AppealsService);
* Получение обращения по ID (AppealsService);
* Обновление данных по обращению (AppealsService);
* Загрузка изображений (ApiGatewayService);
* Выгрузка (просмотр) изображений по ID (ApiGatewayService);
* Отправка письма на электронную почту с вложением (EmailService);
* Отправка письма на электронную почту без вложения (EmailService);

Для примера рассмотрим запрос на создание пользователя. Спецификация REST API требует того, чтобы при операциях создания использовался HTTP метод POST. Тело запроса состоит из полей, которые необходимы для создания пользователя. При успешном создании пользователя возвращается код 200 и объект, описывающий созданного пользователя (который включает в себя поля, переданные при создании и новые поля, добавленные после создания). В случае ошибки выдается код 400 и сообщение «Bad Request».

Листинг 1. Запрос на создание пользователя:

Метод: POST

URI: /user

Тело запроса:

*{*

*"created\_at": "2022-18-05 10:09:21.61",*

*"login": “AlexRudenkiy777”,*

*"name": “Александр”,*

*"surname": “Руденький”,*

*"patronymic": “Олегович”,*

*"passwordHash": “qwerty”,*

*"email": “alex-rudenkiy@mail.ru”*

*}*

Ответ: 200 OK

*{*

*"created\_at": "2022-18-05 10:09:21.61",*

*"login": “AlexRudenkiy777”,*

*"name": “Александр”,*

*"surname": “Руденький”,*

*"patronymic": “Олегович”,*

*"passwordHash": “d8578edf8458ce06fbc5bb76a58c5ca4”,*

*"email": “alex-rudenkiy@mail.ru”*

*}*

* 1. **Структура базы данных**

При разработке микросервисов была использована СУБД PostgreSQL. Далее представлено описание структур разработанных БД.

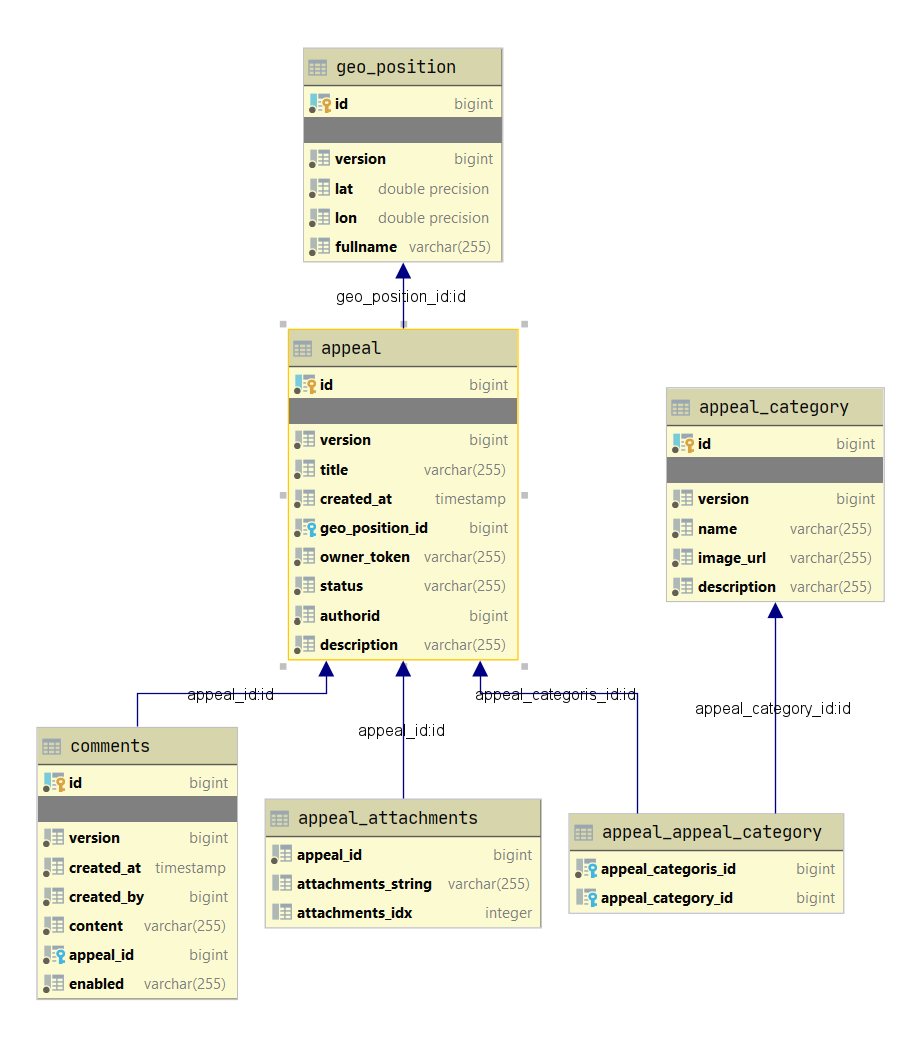


Рисунок 2.7. Структура базы данных «Обращения» (относится к микросервису Appeals)

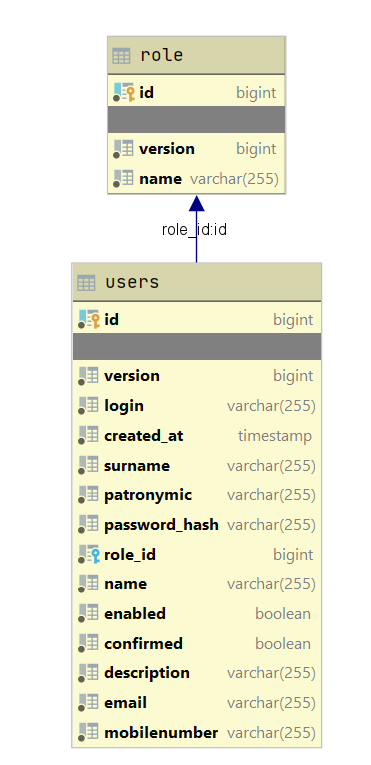


Рисунок 2.8. Структура базы данных «Пользователи» (относится к микросервису Users).

* 1. **Тестирование**

Для тестирования функциональности системы составим тест-кейс таблицу, в которой составим возможные сценарии работы с платформой и отметим какие прошли успешно, а какие нет.

|  |  |
| --- | --- |
| Название тест-кейса и детали проверки | Результат проверки |
| Посещение страницы “Главная страница”/“Архив проблем”/“Настройки”/“Регистрация”/“Авторизация”/ “Страница обращения” | + |
| Выполнение регистрации/авторизация пользователя | + |
| Выполнение регистрации с не валидными данными (проверка механизма валидации и появления соответствующего всплывающего сообщения с информацией об ошибке) | + |
| Выполнение регистрации/авторизации при условии, что сервер не отвечает (проверка появления соответствующего всплывающего сообщения с информацией об ошибке и редиректа на страницу профиля пользователя) | + |
| Создание обращения (проверка механизма валидации и корректной работы соответствующего всплывающего сообщения с информацией об ошибке) | + |
| Просмотр конкретного обращения в “Архиве проблем” (проверка работы модального окна) | + |
| Смена логина, ФИО пользователя на странице “Настройки” | + |
| Установка роли пользователя в настройках пользователя (проверка производится в роли Модератора) | + |
| Обновление содержимого текста произвольного обращения на “индивидуальной странице обращения” (проверка производится в роли Модератора) | + |
| Смена статуса произвольного обращения на “индивидуальной странице обращения” (проверка производится в роли Модератора и частично в роли Представителя администрации, проверяется механизм формирования pdf документов и их отправка представителю(-ям) администрации) | + |
| Создание комментариев под конкретным обращением на странице “архив проблем” | + |
| Создание комментариев на странице “индивидуальная страница обращения” | + |
| Удаление произвольного комментария на странице “архив проблем” и “индивидуальная страница обращения” (проверка производится в роли Модератора) | + |
| Удаление произвольного пользователя на странице “участники” (проверка производится в роли Модератора) | + |

В ходе проверки тест-кейсов, было проверено корректное взаимодействие всех микросервисов и то, что все сценарии были выполнены успешно.

**Заключение**

Данная курсовая работа является результатом систематизации знаний, полученных при исследовании вопросов о подходе к проектированию и оптимизации серверного ПО.

В первой части данной работы была кратко описана предметная область, существующие разработки, а также в ходе сравнительного анализа были выявлены их достоинства и недостатки выбранных решений. По итогу было сформировано описание разрабатываемой системы, её назначение и все необходимые требования для её проектирования и будущей реализации.

В конструкторском разделе, были подробно разобраны все этапы проектирования разрабатываемой микросервисной архитектуры, определенны механизмы взаимодействия микросервисов между собой, а также описаны на разных уровнях с помощью Use-Case диаграмм.

В аналитическом разделе, приведены основные сведения разработанной системы, в частности описание REST API для каждого из микросервисов, а также структуры баз данных, которые им соответствуют. Также в конце данного раздела были приведены основные сценарии для проведения тестирования разрабатываемой системы.

В рамках данной курсовой работы была подробно изучена литература на тему архитектурных подходов разработки системы, в частности принципы и технологии разработки микросервисных архитектур. В качестве результата проведённой работы был разработан портал, для сбора и обработки обращений от жителей определённого города.