Тема индивидуального задания: «Информационный портал сети досуговых центров».

# Техническое задание

# Глоссарий

1. «Горячее» переконфигурирование системы - способность системы применять изменения без перезапуска и перекомпиляции.
2. Валидация - проверка данных на соответствие заданным условиям и ограничениям.
3. REST - архитектурный стиль взаимодействия компонентов распределённого приложения в сети.
4. Досуговый центр (Центр) – место где проводятся занятия
5. Элемент расписания – статья, содержащая время и дату начала занятия в центре
6. Новость – статья, содержащая время и дату новости в центре
7. Медиана времени отклика - среднее время предоставления данных Пользователю.
8. Учетная запись – информация, хранящаяся в базе данных пользователей о пользователе и его роли на портале
9. Роль – характеристика учетной записи, предоставляющая ему различные права на портале
10. Права – возможности действий пользователя на портале

# Основания для разработки

# **Назначение разработки**

Главное назначение разрабатываемого портала - предоставление (целевая аудитория и зачем это надо) пользователю возможности просмотра информации о досуговом центре и получения информации в контексте досугового центра о расписании его занятий и новостей. В контексте центра пользователь должен иметь возможность создать запись на интересующее его занятие. Администратор должен иметь возможность создавать и редактировать новости и элементы в расписании. Суперпользователь должен иметь возможность добавлять досуговые центра на портал и редактировать информацию о любых центрах. Администратор использует элементы расписания и новости для привлечения пользователей на занятия. Все пользователи портала просматривают портал для личных целей.

# Существующие аналоги

Среди аналогов разрабатываемого проекта можно отметить такие порталы, как 2gis.ru, zoon.ru, yell.ru. Данный проект должен иметь следующие преимущества перед существующими аналогами:

1. Предоставление расписания для каждого досугового центра
2. Предоставление новостей для каждого досугового центра
3. Предоставление возможности записаться на занятие для каждого досугового центра

# Описание системы

Проект должен представлять собой портал предоставление информации заинтересованным лицам, какая целевая аудитория.

Пользователь может выступать в качестве администраторов, создающих расписания и новости и в качестве суперпользователей, добавляющих новые досуговые центры в сеть и редактирующий информацию о них. Администраторы имеют возможность получать список пользователей, записавшихся на занятия.

# Общие требования к системе

1. Система должна поддерживать возможность «горячего» переконфигурирования системы. Необходимо поддерживать возможность добавления нового центра, элемента расписания, новости и записи на занятие во время работы системы без рестарта;
2. Время восстановления системы после сбоя не должно превышать 15 минут;
3. Каждый узел должен автоматически восстанавливаться после сбоя;
4. Обеспечить безопасность работоспособности системы за счет отказоустойчивости узлов;

# Требования к функциональным характеристикам

1. Медиана времени отклика системы на запросы пользователя на получение информации не должна превышать 3 секунд
2. Медиана времени отклика системы на действия пользователя должна быть менее 800мс при условии работы на рекомендованной аппаратной конфигурации, задержках между взаимодействующими сервисами менее 200мс и одновременном числе работающих пользователей менее 100 на каждый сервер, обслуживающий внешний интерфейс.

# Функциональные требования к порталу с точки зрения пользователя

Портал должен обеспечивать реализацию следующих функций:

1. Система должна обеспечивать регистрацию и авторизацию пользователей с валидацией вводимых данных через интерфейс приложения.
2. Система должна обеспечивать аутентификацию пользователей.
3. Система должна обеспечивать разделение пользователей на четыре роли:
   1. Неавторизированный пользователь; роль
   2. Пользователь;
   3. Администратор;
   4. Суперпользователь
4. Система должна предоставлять любому пользователю, в том числе и неавторизированному, следующие функции:
   1. Просмотр информации о досуговых центрах в сети
   2. Просмотр расписаний досуговых центров
   3. Просмотр новостей досуговых центров
5. Система должна предоставлять Пользователю с ролью пользователь следующие функции:
   1. Запись на занятие в досуговом центре
   2. Функции всех, в том числе и неавторизированных пользователей
6. Система должна предоставить посльзователю с ролью администратор следующие функции (сделать для каждой роли отдельно в начале этого пункта)
7. Система должна предоставлять Пользователю с ролью администратор следующие функции:
   1. Создание элемента расписания в досуговом центре
   2. Создание новости в досуговом центре
   3. Удаление элемента расписания
   4. Удаление новости
   5. Функции пользователя с ролью пользователь
8. Система должна предоставлять Пользователю с ролью суперпользователь следующие функции:
   1. Создание нового досугового центра
   2. Редактирование информации о досуговом центре
   3. Функции пользователя с ролью администратор
   4. Выдача ролей другим пользователям
   5. Получение информации о пользователях

## Входные данные Пользователя:

* Имя, не более 256 символов, уникальное
* Логин, не более 256 символов
* Пароль, не более 256 символов

## Выходные данные:

Выходными данными системы являются веб страницы. В зависимости от роли и запроса пользователя они содержат: ОДНА ВЕБ СТРАНИЦА КОТОРАЯ ОБНОВЛЯЕТСЯ КЕМ С КАКОЙ ПЕРЕДИЧНОСТЬЮ (дописать)

* Список досуговых центров и информацию о них
* Список новостей в центрах и расписаний в них
* Информацию об аккаунте пользователя
* Интерфейсы добавления, удаления, редактирования и просмотра:
  + Досугового центра
  + Информации о досуговом центре
  + Новостей
  + Элементов расписаний
  + Записей на занятия
  + Информации о пользователях
* И другие.

# Требования к программной реализации

1. Требуется использовать СОА (сервис-ориентированную архитектуру) для реализации системы;
2. Система состоит из микросервисов. Каждый микросервис отвечает за свою область логики работы приложения;
3. Взаимодействие между сервисами осуществляется посредством HTTP-запросов
4. Данные сервисов должны храниться в базе данных. Каждый сервис взаимодействует только со своей схемой данных. Взаимодействие сервисов происходит по технологии REST;
5. При недоступности систем портала должна осуществляться деградация функционала или выдача пользователю сообщения об ошибке;
6. Необходимо предусмотреть авторизацию пользователей через интерфейс приложения
7. Для запросов, выполняющих обновление данных на нескольких узлах распределенной системы, в случае недоступности одной из систем, необходимо выполнять полный откат транзакции;
8. Необходимо сохранять профили всех пользователей в базе данных в хешированном виде.
9. Приложение должно поддерживать возможность горизонтального и вертикального масштабирования за счет увеличения количества функционирующих узлов и совершенствования технологий реализации компонентов и всей архитектуры системы

# Топология системы

Топология разрабатываемой системы, изображена на рисунке 1.

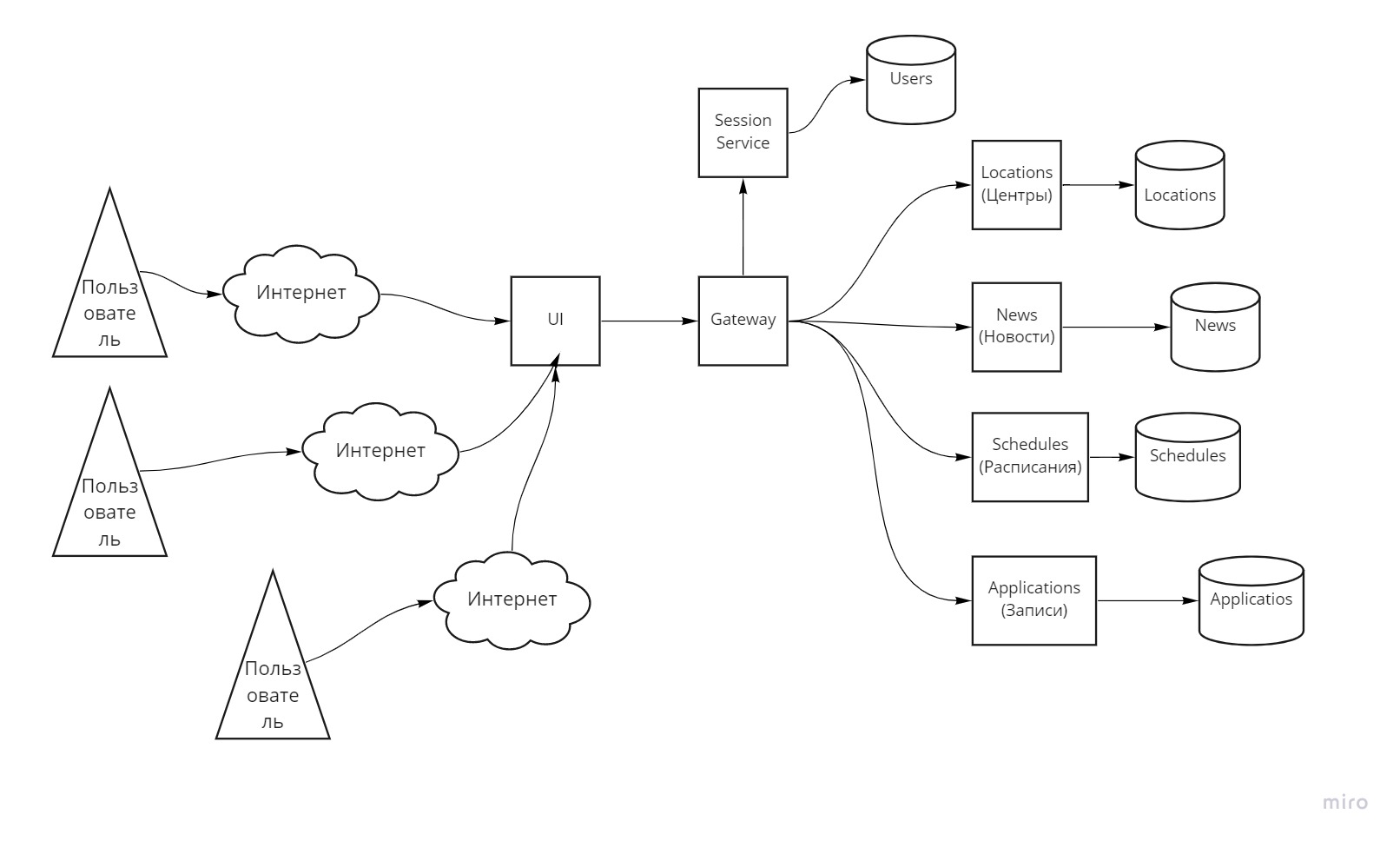


Рисунок 1. Топология системы

Разрабатываемая система состоит из Фронт-энда (UI), Gateway (Сервис - координатор), Сессионного сервиса (Session service) и 4 подсистем:

* Сервис центров (Locations)
* Сервис новостей (News)
* Сервис расписаний (Schedules)
* Сервис записей на занятия (Applications)

Топология системы позволяет визуализировать связи между ее компонентами. НАРИСОВАТЬ ПО ГОСТАМ

# Общие требования к подсистемам

1. Фронт-энд – серверное приложение, при разработке которого следует учесть следующие нюансы:
   1. Фронт-энд должен принимать запросы по протоколу HTTP и формировать ответ пользователю в формате HTML-страниц;
   2. Фронт-энд является посредником между пользователями, передавая их запросы последовательно на сервис агрегации запросов;
   3. В соответствии с выбранными технологиями реализации целесообразно использовать библиотеку JQuery, языки HTML, CSS и JS, для создания верстки HTML-страниц проекта.
2. реализации бэк-эндов должны быть предъявлены следующие требования:
   1. Прием и возврат данных должен происходить в формате JSON по протоколу HTTP;
   2. Если результаты работы сервиса необходимо сохранять в базе данных, то доступ к ней должен осуществляться по протоколу HTTP. Доступ к базе данных может осуществляться только из подсистем, работающих напрямую с данными ее таблиц.

# Функциональные требования к сервисам

1. Сервис агрегации запросов - предоставляет пользовательский интерфейс и внешний API системы.

Сервис должен реализовывать следующий функционал:

* Проверка существования пользователя
* Регистрация пользователя
* Аутентификация пользователя
* Получение информации о всех центрах
* Получение новостей для одного центра
* Получение расписания для одного центра
* Создание и удаление новости (только для определенных ролей)
* Создание и удаление элемента расписания (только для определенных ролей)hjkm
* Создание, удаление и редактирование одного центра (только для определенных ролей)
* Просмотр информации о пользователях (только для определенных ролей)
* Выдача ролей пользователю (только для определенных ролей)
* Просмотр записей на занятия (только для определенных ролей)

1. Сервис работы с центрами – отвечает за хранение информации о досуговых центрах и ее редактирование

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* Название центра
* Информация о центре
* Адрес центра

Сервис должен реализовывать следующий функционал:

* Получение всей информации о центрах
* Редактирование всей информации о центре (только для определенных ролей)
* Удаление центра (только для определенных ролей)

1. Сервис работы с новостями – отвечает за хранение новостей и ее редактирование

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* Содержание новости
* Дата создания новости
* Время создания новости
* Название центра, к которому принадлежит новость

Сервис должен реализовывать следующий функционал:

* Добавление новости (только для определенных ролей)
* Удаление новости (только для определенных ролей)

1. Сервис работы с расписаниями – отвечает за хранение информации о расписаниях и ее редактирование

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* Содержание элемента расписания
* Дата проведения элемента расписания
* Время проведения элемента расписания
* Название центра, к которому принадлежит элемент расписания

Сервис должен реализовывать следующий функционал:

* Добавление элемента расписания (только для определенных ролей)
* Удаление элемента расписания (только для определенных ролей)

1. Сервис работы с записями на занятия – отвечает за хранение информации о записях на занятия и ее редактирование

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* ID элемента расписания
* ID пользователя

Сервис должен реализовывать следующий функционал:

* Добавление записи на занятие
* Удаление записи на занятие (только для определенных ролей)
* Просмотр всех записей на занятия (только для определенных ролей)

1. Сервис регистрации и авторизации пользователей

Хранимая в базе данных сущность, ассоциированная с сервисом, имеет следующие обязательные поля:

* Логин
* Пароль
* Имя
* Роль

Сервис должен реализовывать следующий функционал:

* Идентификация и аутентификация пользователя
* Выдача токена
* Регистрация пользователя
* Выдача прав пользователю (только для определенных ролей)
* Получение информации и о всех пользователях (только для определенных ролей)

# Требования к надежности и к документации

Система должна работать в соответствии с данным техническим заданием без перезапуска

# Конструкторский раздел

# Концептуальный дизайн

Концептуальный дизайн системы содержит наиболее общие схемы описания функционала приложения с точки зрения пользователей [2]. Одной из таких схем является IDEF0-модель и графические модели, входящие в нее [4]. На рисунке 2 отображена контекстная диаграмма верхнего уровня, которая обеспечивает наиболее общее или абстрактное описание работы системы. Данный вид диаграммы позволяет формализовать описание запросов пользователя и ответов системы на данные запросы, отобразив систему в виде “черного” ящика.

Диаграммы переделать

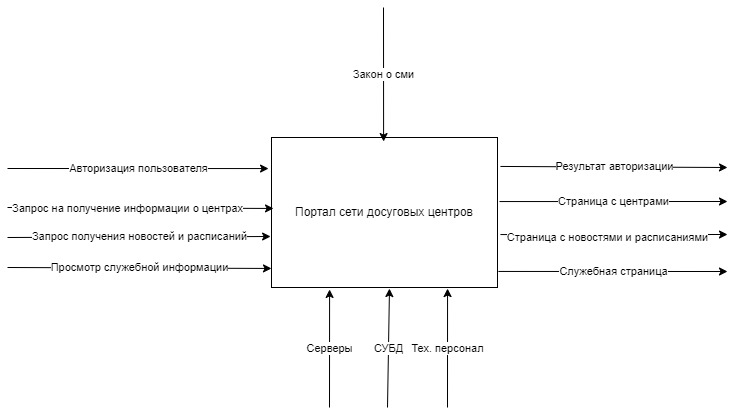


Рисунок 2. Концептуальная модель системы в нотации IDEF0.

Для уточнения деталей работы системы применяется декомпозиция функций, отображенных на диаграмме верхнего уровня, при помощи создания дочерних диаграмм. В качестве примера на рисунке 3 изображена дочерняя диаграмма, которая определяет последовательность выполнения операций в системе при обработке запроса администратора на получение всех записей на занятия.

пределать

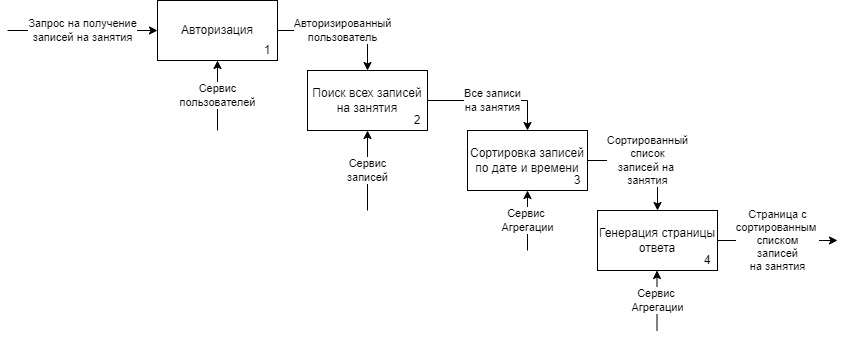


Рисунок 3. Детализированная концептуальная модель системы в нотации IDEF0.

Детализированная диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский блок, но описывает ее более подробно. Поэтому такая диаграмма может быть создана для любого из запросов, отображенных на диаграмме верхнего уровня. Каждый из блоков детализированной диаграммы может быть в свою очередь также описан при помощи дочерней диаграммы.

# Сценарии функционирования системы

Сценарии функционирования или использования системы описывают конкретную последовательность действий, иллюстрирующую поведение пользователя при работе с приложением. Далее приведены подробные сценарии основных возможных действий пользователя.

## Регистрация пользователя:

1. Пользователь вводит логин, пароль и имя в интерфейсе приложения;
2. Пользователь нажимает кнопку «Регистрация», тем самым подтверждая верность своих данных, а также согласие на их обработку и хранение;
3. Если пользователь с введенным для регистрации именем уже существует или не вводит имя или не вводит пароль, то клиент получает сообщение об ошибке. При успешной регистрации пользователь получает сообщение об успешной регистрации.

## Авторизация пользователя:

1. Пользователь вводит логин и пароль в интерфейсе приложения;
2. Пользователь нажимает кнопку «Войти» в интерфейсе приложения;
3. При обнаружении ошибки в данных, пользователь получает сообщение об ошибке. При совпадении данных с записью в базе данных пользователей пользователь получает доступ к системе с правами, полученными из значения его роли, получает сообщение об успехе

## Получение информации о досуговых центрах:

1. При переходе пользователя на главную страницу приложения, автоматически Сервис Агрегатор достучаться до базы с центрами из базы данных получает всю информация о центрах;
2. Фронт-энд возвращает пользователю веб страницу из полученных данных

## Получение новостей и расписания занятий для досугового центра:

1. Пользователь нажимает на интересующий его центр
2. Из базы данных новостей Сервис Агрегатор получает новости для интересующего центра
3. Из базы данных расписаний Сервис Агрегатор получает расписание для интересующего центра
4. Фронт-энд возвращает пользователю страницу с полученными данными

## Создание или изменение информации о досуговом центре:

1. Пользователь нажимает кнопку «Центры»
2. Фронт-энд генерирует страницу с выбором действий
   1. Пользователь вводит данные нового центра и нажимает кнопку «создать»
      1. Сервис Агрегатор создает новый центр
      2. Фронт-энд сообщает пользователю об успехе или неудаче в процессе создания
   2. Пользователь вводит ID центра на удаление и нажимает кнопку «удалить»
      1. Сервис Агрегатор удаляет центр
      2. Фронт-энд сообщает пользователю об успехе или неудаче в процессе удаления
   3. Пользователь вводит данные для обновления центра и нажимает кнопку «изменить»
      1. Сервис Агрегатор обновляет данные для центра
      2. Фронт-энд сообщает пользователю об успехе или неудаче в процессе обновления данных

## Получение записей на занятия:

1. Пользователь нажимает кнопку «записи»
2. Сервис Агрегатор получает все записи на занятия
3. Сервис Агрегатор сортирует все записи на занятия
4. Фронт-энд возвращает пользователю страницу со всеми записями на занятия

## Создание или удаление новости

1. Пользователь нажимает кнопку «новость»
2. Фронт-энд генерирует страницу с выбором действий
   1. Пользователь вводит данные новой новости и нажимает кнопку «создать»
      1. Сервис Агрегатор создает новость
      2. Фронт-энд сообщает пользователю об успехе или неудаче в процессе создания
   2. Пользователь вводит ID новости и нажимает кнопку «удалить»
      1. Сервис Агрегатор удаляет новость
      2. Фронт-энд сообщает пользователю об успехе или неудаче в процессе удаления

## Создание или удаление элемента расписания:

1. Пользователь нажимает кнопку «расписание»
2. Фронт-энд генерирует страницу с выбором действий
   1. Пользователь вводит данные нового элемента расписания и нажимает кнопку «создать»
      1. Фронт-энд сообщает пользователю об успехе или неудаче в процессе создания
   2. Пользователь вводит ID элемента расписания и нажимает кнопку «удалить»
      1. Сервис Агрегатор удаляет новость
      2. Фронт-энд сообщает пользователю об успехе или неудаче в процессе удаления

# Диаграммы прецедентов

Графически сценарии функционирования системы можно представить при помощи диаграмм прецедентов. Они позволяют схематично отобразить типичные сценарии взаимодействия между клиентами и приложением. В системе выделены 4 основных роли: неавторизированный пользователь, пользователь, администратор и суперпользователь диаграммы прецедентов для этих ролей изображены на рисунках 4, 5 и 6

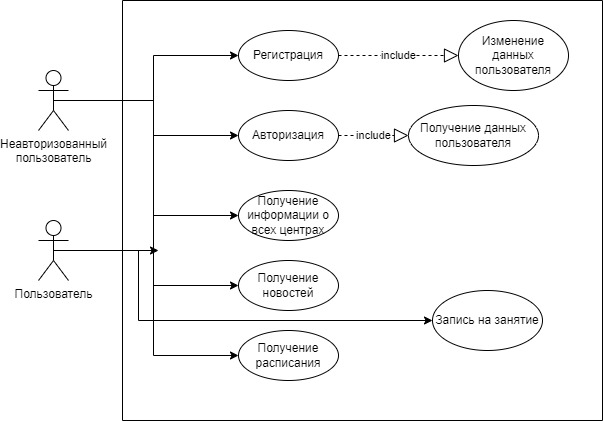


Рисунок 4. Диаграмма прецедентов с точки зрения пользователя и неавторизованного пользователя.

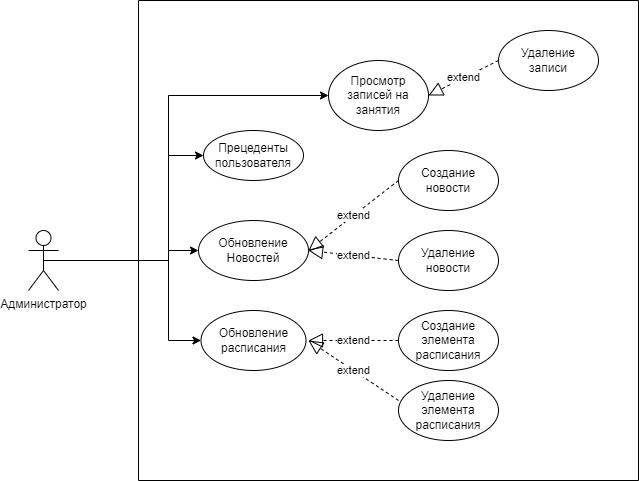


Рисунок 5. Диаграмма прецедентов с точки зрения администратора.

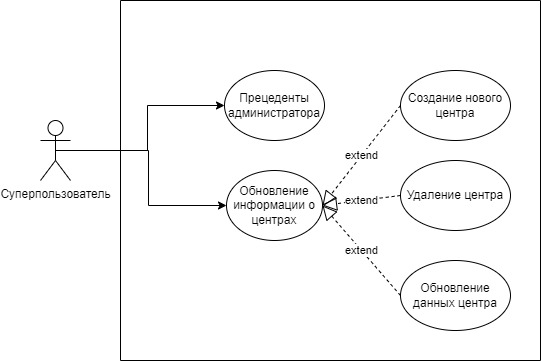


Рисунок 6. Диаграмма прецедентов с точки зрения суперпользователя.

Не создание центра а «информации»

# Спецификации сценариев

Приведенные сценарии могут иметь как основной поток выполнения, который выполняется чаще всего, так и альтернативные потоки, описывающие выполнение запроса при отклонении от основного хода сценария. Все возможные ходы выполнения сценария описываются при помощи спецификаций. Примеры спецификаций для описанных выше сценариев приведены в данном разделе.

## Спецификация сценария регистрация пользователя

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь вводит логин и пароль, нажимает кнопку «Вход» | Пользователь получает сообщение об успехе |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь вводит логин и пароль, нажимает кнопку «Вход» | Пользователь получает сообщение об ошибке |

## Спецификация сценария регистрация пользователя:

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь вводит логин, пароль и имя в интерфейсе приложения, нажимает кнопку «Регистрация» | Пользователь получает сообщение об успехе |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь вводит логин, пароль и имя в интерфейсе приложения, нажимает кнопку «Регистрация» | Пользователь получает сообщение об ошибке |

## Спецификация сценария получение информации о досуговых центрах:

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь переходит на главную страницу приложения | Пользователь получает веб страницу с информацией о досуговых центрах |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь переходит на главную страницу приложения | Пользователь получает сообщение об ошибке |

## Спецификация сценария получение новостей и расписания занятий для досугового центра:

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает на интересующий его центр | Пользователь получает веб страницу с информацией о центре |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает на интересующий его центр | Пользователь получает сообщение об ошибке |

## Спецификация сценария создание или изменение информации о досуговом центре:

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает на кнопку «Центры» | Пользователь получает веб страницу с выбором действий |
| Пользователь вводит данные нового центра и нажимает кнопку «создать» | Пользователь получает сообщение об успехе |
| Пользователь вводит ID центра на удаление и нажимает кнопку «удалить» | Пользователь получает сообщение об успехе |
| Пользователь вводит данные для обновления центра и нажимает кнопку «изменить» | Пользователь получает сообщение об успехе |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает на кнопку «Центры» | Пользователь получает веб страницу с выбором действий |
| Пользователь вводит данные нового центра и нажимает кнопку «создать» | Пользователь получает сообщение об ошибке |
| Пользователь вводит ID центра на удаление и нажимает кнопку «удалить» | Пользователь получает сообщение об ошибке |
| Пользователь вводит данные для обновления центра и нажимает кнопку «изменить» | Пользователь получает сообщение об ошибке |

## Спецификация сценария получение записей на занятия:

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает кнопку «записи» | Пользователь получает веб страницу с информацией о всех записях на занятия |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает кнопку «записи» | Пользователь получает сообщение об ошибке |

## Спецификация сценария создание или удаление новости

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает кнопку «новость» | Пользователь получает веб страницу с выбором действий |
| Пользователь вводит данные новой новости и нажимает кнопку «создать» | Пользователь получает сообщение об успехе |
| Пользователь вводит ID новости и нажимает кнопку «удалить» | Пользователь получает сообщение об успехе |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает кнопку «новость» | Пользователь получает веб страницу с выбором действий |
| Пользователь вводит данные новой новости и нажимает кнопку «создать» | Пользователь получает сообщение об ошибке |
| Пользователь вводит ID новости и нажимает кнопку «удалить» | Пользователь получает сообщение об ошибке |

## Спецификация сценария Создание или удаление элемента расписания:

|  |  |
| --- | --- |
| **Нормальный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает кнопку «расписание» | Пользователь получает веб страницу с выбором действий |
| Пользователь вводит данные нового элемента расписания и нажимает кнопку «создать» | Пользователь получает сообщение об успехе |
| Пользователь вводит ID элемента расписания и нажимает кнопку «удалить» | Пользователь получает сообщение об успехе |

|  |  |
| --- | --- |
| **Альтернативный ход сценария** | |
| **Действие пользователя** | **Отклик системы** |
| Пользователь нажимает кнопку «расписание» | Пользователь получает веб страницу с выбором действий |
| Пользователь вводит данные нового элемента расписания и нажимает кнопку «создать» | Пользователь получает сообщение об ошибке |
| Пользователь вводит ID элемента расписания и нажимает кнопку «удалить» | Пользователь получает сообщение об ошибке |

# Логический дизайн

В процессе создания концептуального дизайна системы были отражены основные сценарии взаимодействия пользователя и системы. В разделе логического дизайна представлена организация элементов системы и их взаимодействие между собой.

На основе функциональных требований к выделенным подсистемам, а также объектов, о которых необходимо хранить данные в системе, была разработана схема данных приложения. Результат ее проектирования отображен на условной спецификации таблиц базы данных, приведенных на рисунке 7.

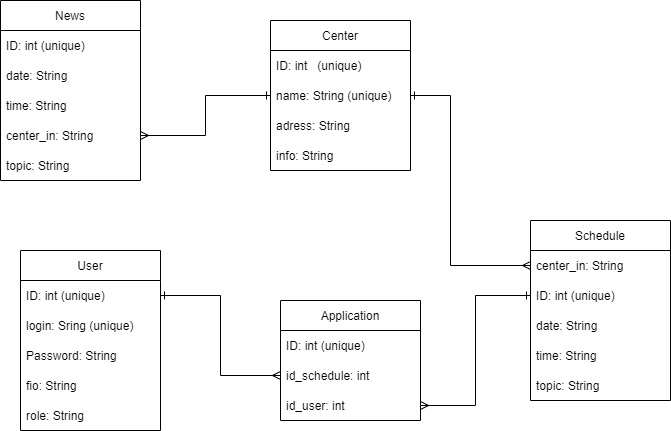


Рисунок 7. Схема базы данных системы.

## Спецификация таблицы User:

Таблица содержит данные о пользователях

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| ID | Int | Идентификатор пользователя |
| login | String | Логин пользователя |
| password | String | Хешированный пароль пользователя |
| fio | String | Имя пользователя |
| role | String | Роль пользователя |

## Спецификация таблицы Center:

Таблица содержит данные о центрах

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| ID | Int | Идентификатор центра |
| name | String | Название центра |
| adress | String | Адрес центра |
| info | String | Информация о центре |

## Спецификация таблицы News:

Таблица содержит данные о новостях

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| ID | Int | Идентификатор новости |
| date | String | Дата новости |
| time | String | Время новости |
| center\_in | String | Названия центра в котором была новость |
| topic | String | Текст новости |

## Спецификация таблицы Schedule:

Таблица содержит данные о элементах расписаний

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| ID | Int | Идентификатор элемента расписания |
| center\_in | String | Названия центра в котором находится элемент расписания |
| date | String | Дата элемента расписания |
| time | String | Время элемента расписания |
| topic | String | Текст элемента расписания |

## Спецификация таблицы Application:

Таблица содержит данные о записях на занятия

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип атрибута** | **Описание атрибута** |
| ID | Int | Идентификатор записи на занятие |
| id\_schedlue | Int | Идентификатор элемента расписания |
| id\_user | Int | Идентификатор пользователя |

# Структура сервиса

На основе разработанной схемы данных можно установить соответствие сущностей и сервисов, описанных в предыдущем разделе. Например, для работы с сущностью Centers был выделен сервис центров. Рассмотрим его устройство более подробно. Для каждого сервиса должен быть создан класс контроллер, а для каждого сервера должен быть создан роутер, использующий свой контроллер. Контроллер позволяет получать информацию из базы данных при помощи объекта класса sequelize (ORM). В качестве примера на рисунке 8 представлена диаграмма класса контроллера для сервиса центров.

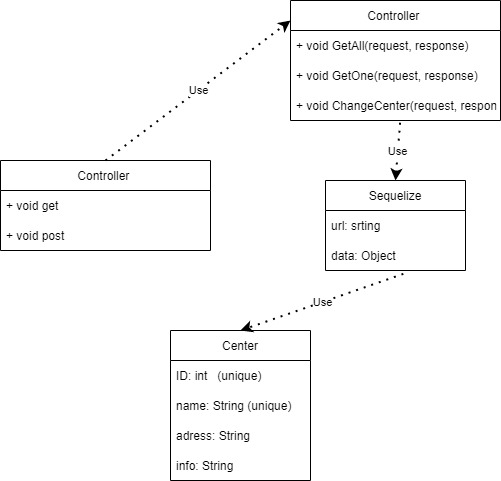


Рисунок 8. Диаграмма классов сервиса центров.

Функциональные требования, предъявляемые к сервису публикаций, реализуются при помощи методов контроллера Controller. Далее приведено описание каждого метода данного контроллера.

## Спецификация класса Controller

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| GetAll | Оправляет всю информацию по всем центрам |
| GetOne | Отправляет информацию по конкретному центру |
| ChangeCenter | Изменяет информацию для выбранного центра |

# Диаграммы последовательности действий

Для описания поведения компонентов системы на единой оси времени используются диаграммы последовательности действий, при помощи которых можно описать последовательность действий для каждого прецедента, необходимую для достижения цели. Например, на рисунке 9 изображен процесс изменения данных для центра.

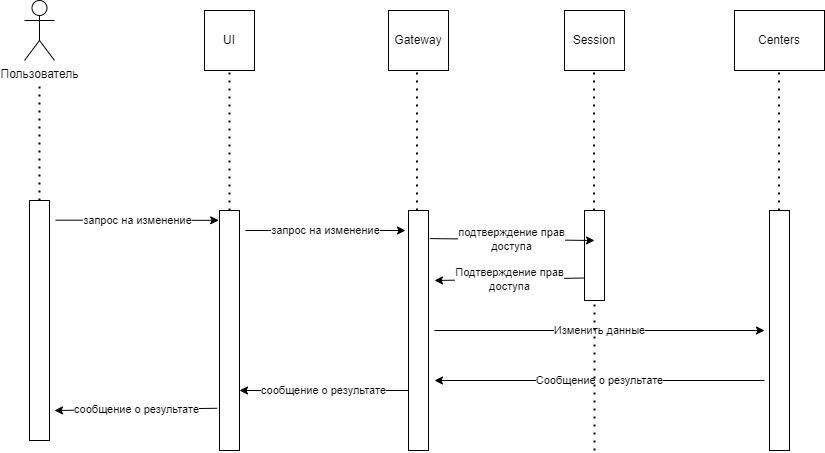


Рисунок 9. Диаграмма последовательности действий при изменении данных для центра

Пользователь отправляет запрос на получение списка записей на занятия на фронт-энд, фронт-энд отправляет запрос на сервис Агрегатор. Он в свою очередь запрашивает у сессионного сервиса разрешения на просмотр данной информации и получив ответ запрашивает у сервиса записей нужные данные. Возвращает их на фронт-энд, который возвращает пользователю веб страницу с полученными данными.

# 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 10, отображает модель информационной системы с точки зрения хранения, передачи и обработки данных во время обработки запроса пользователя на изменение данных для центра.

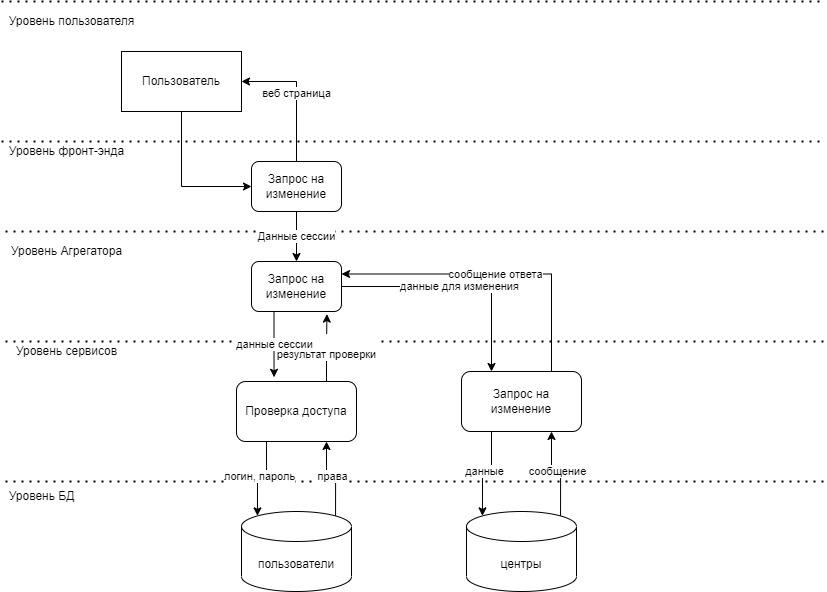


Рисунок 10. Диаграмма потоков данных при запросе пользователем изменения данных в центре.

# Архитектура системы

Основополагающей идеей построения программной архитектуры является идея снижения сложности системы путём абстракции и разграничения полномочий. В данном проекте каждая функциональная область реализована посредством собственного микросервиса. Этот подход позволяет бороться со сложностью современных систем. Архитектура системы призвана показать способ развертывания системы во внешних средах. На рисунке 11 представлена архитектура системы, которая показывает размещение элементов системы на физических носителях и способах их взаимодействия, то есть, указаны протоколы, по которым происходит информационный обмен.

Клиент заходит через браузер, рисунок по госту

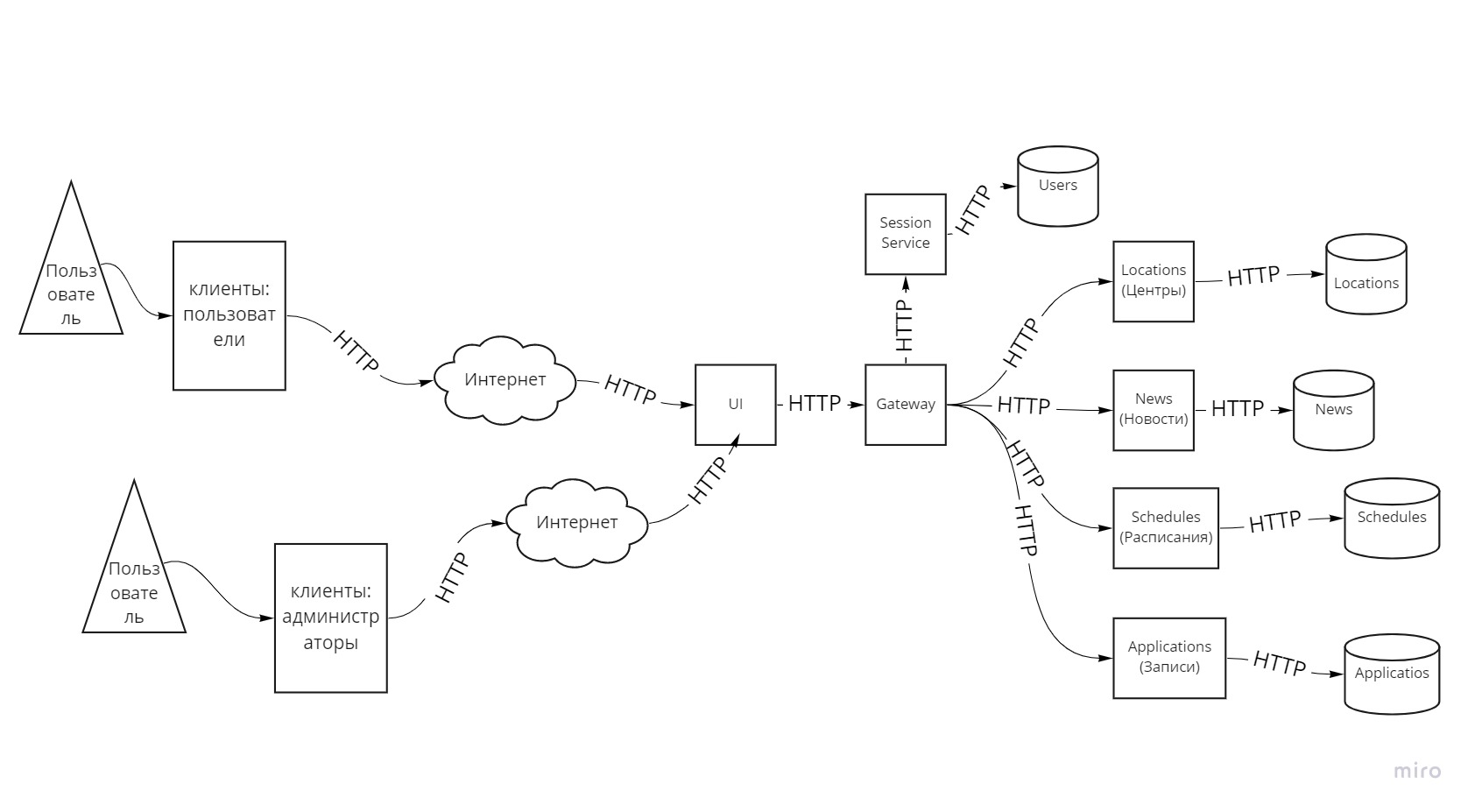


Рисунок 11. Архитектура разрабатываемой системы.

# Технологический раздел

# Выбор языка и фреймворка для разработки системы

Для реализации системы использовалась сервис-ориентированная архитектура (СОА) [5]. Архитектурным стилем взаимодействия между сервисами выбран REST (Representational State Transfer — «передача состояния представления») [6]. В качестве языка реализации серверной части приложения выбран JavaScript, так как он является одним из ведущих языков в разработке информационных систем на сегодняшний день, а также обладает всеми необходимыми особенностями для реализации СОА и REST.

Для системы выбран фрейморк Node.js, Express широко используемый для решения подобных задач, обладающий возможностью запуска на разных платформах и давно используется в разработке.

# Выбор СУБД

Для работы с базами данных была выбрана библиотека Sequelize – ORM поддерживающая все передовые СУБД. Благодаря этому в процессе работы можно заменить СУБД на более подходящее. В качестве основной была выбрана Postgres как оптимальный вариант в связи с широким знакомством с ней в рамках курса «Базы данных».

# Реализация сервисной архитектуры

## Общая архитектура сервиса

Все убрать из кодов

Для соответствия стилям СОА и REST необходимо разбить систему на отдельные сервисы и организовать взаимодействие между ними через интерфейсы, без прямого обращения. Каждый сервис создается как веб-приложение Node.js Express.

Основным элементом сервисов являются роутеры, которые в зависимости от маршрута вызывают определенный метод контроллера.

Роутер сервиса центров представлен в листинге 1.

import { Router } from 'express'

import Controller from './Controller.js'

import middleWare from './middleWare.js'

const router = new Router()

router.get('/testmiddle', middleWare.autorizeSuperuser, middleWare.printLol)

router.post('/test', Controller.test)

router.post('/create', middleWare.autorizeSuperuser, Controller.createCenter)

router.get('/', Controller.getAll)

router.put('/change', middleWare.autorizeSuperuser , Controller.changeCenter)

router.get('/find', Controller.getOne)

export default router

Листинг 1. Роутер сервиса центров.

В роутерах удобно использовать middleware функции, которых нет в основном контроллере, для усиления разделения логики и масштабируемости всей системы.

В листинге 1. Можно увидеть, что перед методами контроллера вызываются middleware функции.

Для подключения к базам данных используется ORM Sequelize. Пример подключение к базе данных показан в листинге 2.

async function authInDB(adress){

    const sequelize = new Sequelize(adress)

    try {

        await sequelize.authenticate()

        console.log('Соединение с БД было успешно установлено')

    } catch (e) {

        console.log('Невозможно выполнить подключение к БД: ', e)

        return 0

    }

    const Center = defineShema(sequelize)

    await Center.sync({ alter: true })

    return {seq: sequelize, modl: Center}

}

const seqAndModel = await authInDB('postgres://postgres:postgres@db\_centers/postgres')

Листинг 2. Подключение к БД центров.

## Взаимодействие между сервисами

Взаимодействие между сервисами осуществляется по протоколу HTTP. Для создания запросов была использована библиотека AXIOS обладающая удобным в применении асинхронным интерфейсом. Пример запроса через AXIOS в листинге 3.

await axios({

            url:'http://auth\_server/auth/premissions',

            method: 'get',

            headers:{

                'authorization': req.headers.authorization

            }

        })

        .then((response) =>{

            let roles = ['SUPERUSER', 'ADMIN']

            let myRole = response.data['role']

            if (response.status == 200){

                if(roles.indexOf(myRole) != -1){

                    next()

                }

                else{

                    res.status(400).json({errorMessage: 'недостаточно прав'})

                }

            }

        })

        .catch((error) =>{

            res.status(400).json({errorMessage: 'не авторизован'})

        })

Листинг 3. Пример асинхронного запроса.

## Реализация отказоустойчивости

Отказоустойчивость обеспечена автоматическим перезапуском приложения. Для этого в файле сборки docker-compose для каждого сервиса установлена инструкция restart: always, что обеспечивает его перезапуск (сервиса)

## Реализация авторизации в системе

Для авторизации и регистрации в системе в базе данных хранятся логины и пароли пользователей. При это пароли хранятся в базе данных в хешированном виде, во избежание утечек данных. Хеширование реализовано при помощи библиотеки bcrypt. Листинг 4 – пример использования функции хеширования для сохранения пароля в базу данных.

let hashedPassword = bcrypt.hashSync(req.body.password, saltRounds);

            //let roleUser = req.body.role ? req.body.role : 'USER'

            createdUser = await User.create({

                login: req.body.login,

                password: hashedPassword,

                fio: req.body.fio

                //role: roleUser

            })

Листинг 4. Хеширование паролей при регистрации.

# Требования к дизайну

# ЛИТЕРАТУРА

1. Ньюмэн С. Создание микросервисов - СПб.: Питер, 2016, - 392 с

2. Техническое задание. Требования к содержанию и оформлению (ГОСТ 19.201- 78). [Электронный ресурс] Режим доступа: URL: <http://internetlaw.ru/gosts/gost/31884/>

3. Вишневская Т.И., Романова Т.Н. Методология программной инженерии: Мет. указания к выполнению лабораторных работ М.: Изд-во МГТУ им. Н.Э. Баумана, 2017.-58 с.

4. Научно-исследовательский Центр CALS - Методология функционального моделирования IDEF0: ИПК Издательство стандартов, 2000 – 75 с.

5. Богданов А., Корхов В., Мареев В. Архитектуры и топологии многопроцессорных вычислительных систем. Изд-во Интуит.Ру, 2013, -176 с.

6. Мартин Клеппман. Высоконагруженные приложения. Программирование, масштабирование, поддержка - СПб.: Питер, 2018.- 640с.