федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский УНИВЕРСИТЕТ информационных технологий, механики и оптики

**Факультет программной инженерии и компьютерной техники**

**Кафедра вычислительной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**ПО ДИСЦИПЛИНЕ «ПРОЕКТИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»**

Выполнил: Ванцев Александр Олегович

Группа: P3400

Преподаватель: Пенской Александр Владимирович

Санкт-Петербург

2019

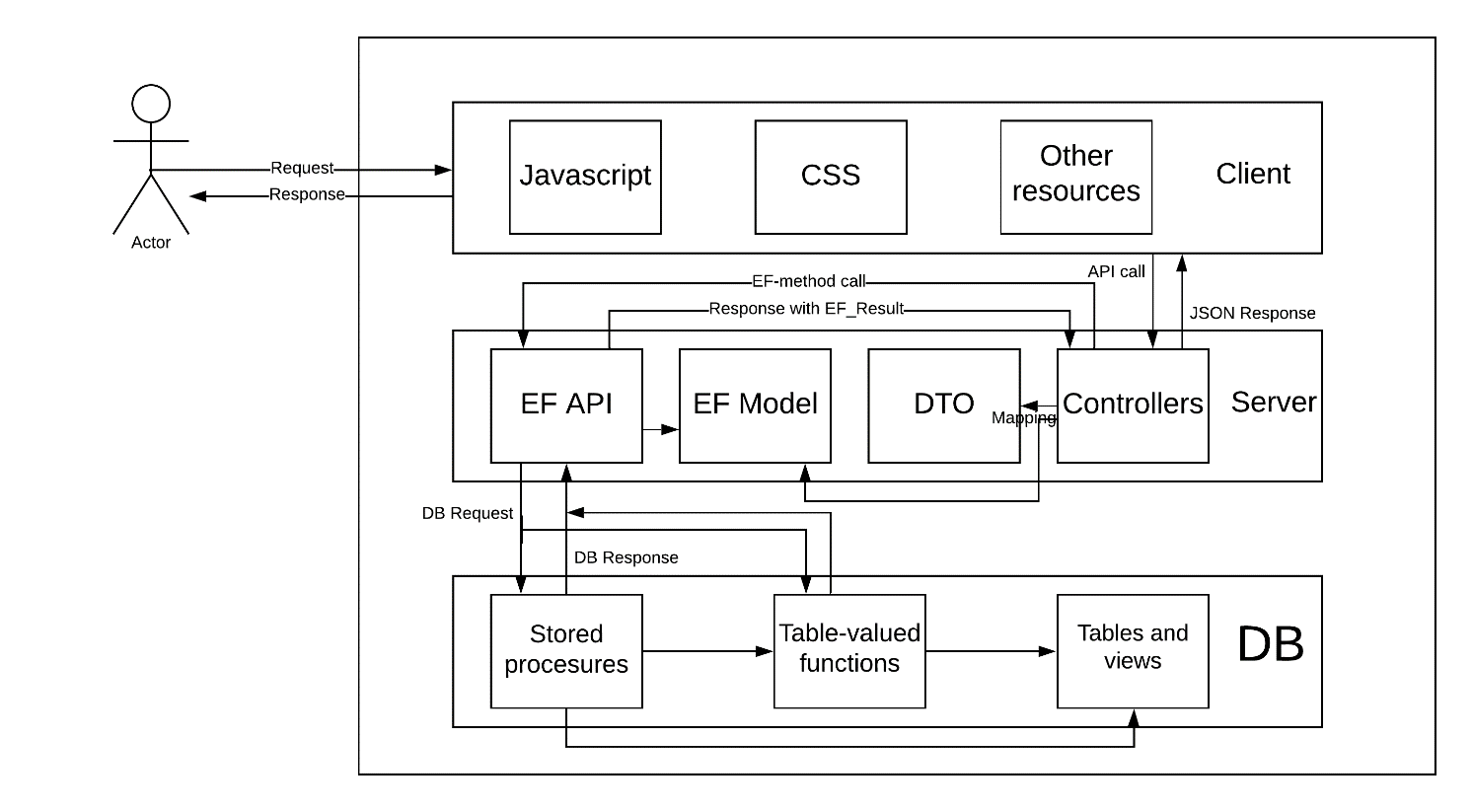
1. Анализ доступных инструментов проектирования.

В качестве инструмента для документирования решений был выбран UML, как наиболее популярный, часто применимый и хорошо разработанный язык проектирования. Большим его преимуществом также является широкая поддержка различными бесплатными средствами проектирования и простота использования, выражающаяся в интуитивно понятном графическом интерфейсе, в отличие от других ADL, требующих специальных инструментов либо изучения нового синтаксиса.

В данной работе были использованы следующие UML-диаграммы:

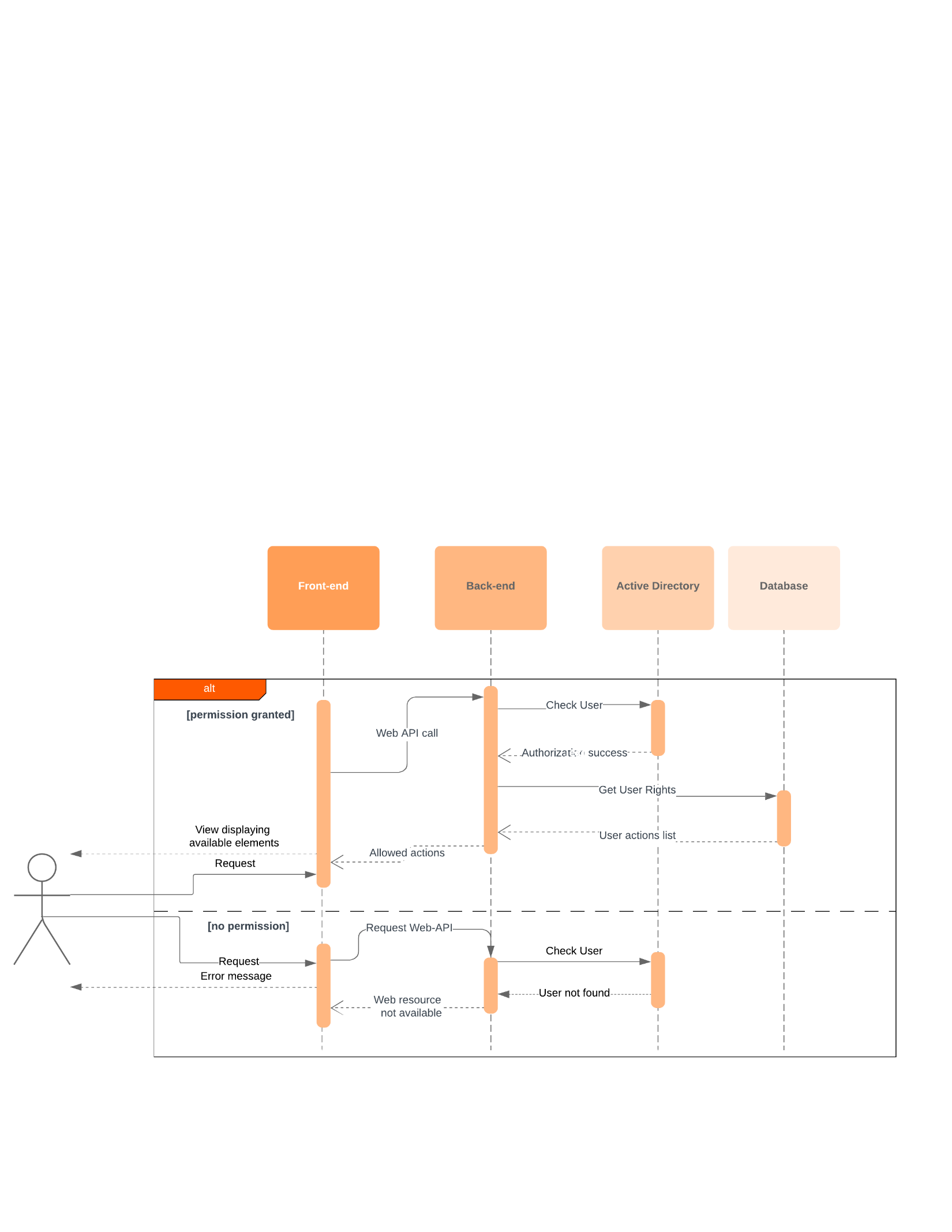
* UML component – для отображения структуры решения, разработанных компонентов, их взаимосвязей и взаимодействий.
* UML sequence – для документирования взаимодействий между компонентами во времени. С помощью данной диаграммы удобно показывать не только какие события происходят в системе, но и их последовательность.
* UML activity – диаграмма деятельности лучше других показывает внутреннюю логику работы системы. В отличие от диаграммы последовательности она нагляднее показывает работу системы, поскольку в ней более удобно реализовано применение условий и циклов.

1. Архитектурные решения проблем:
   1. Разделенное на back- и front-end приложение должно иметь возможность быстрой и простой разработки и добавления новых компонент (страниц) без влияния на существующий код.



Front-end реализуется с использованием Angular 5. Одностраничное приложение обращается к API-методам back-end, написанного на C#. Таким образом достигается малая связанность компонентов, а изменение внутренней работы любого из них никак не отражается на работе другого при условии одинакового ответа сервера на один и тот же набор входных параметров. Back-end по запрашиваемому URL создает экземпляр контроллера и вызывает подходящий метод, обращающийся к базе данных MS SQL с помощью Entity Framework. Использование последнего существенно упрощает архитектуру приложения и разработку новых API-методов, т.к. предоставляет удобный интерфейс для взаимодействия с БД и автоматически генерирует классы-представления данных. Однако поскольку на front-end нужно передавать не все поля возвращаемых данных из базы, применяется auto-mapping для отображения полученных данных на упрощенную (в большинстве случаев) или модифицированную модель.

* 1. Должна обеспечиваться безопасность сайта: ограничение полномочий пользователя в зависимости от его должности.



Данная проблема подразумевает решение двух вопросов: аутентификации и авторизации пользователя на портале.

Первая решается при помощи использования Active Directory. Сервер при получении запроса проверяет, существует ли такой пользователь, и в случае положительного ответа передает управление дальше, в противном случае посылает соответствующий код ошибки.

Если аутентификация пройдена, определяются разрешенные пользователю действия с помощью ролевой модели. Каждому пользователю поставлено в соответствие одна или несколько групп, каждой из которых в свою очередь отвечает определенный набор доступных действий на портале. Происходит запрос к БД, где по ID пользователя определяются его группы и список разрешенных действий, которые в конце концов передаются на front-end. Последний на основе полученных данных определяет, какие элементы нужно отобразить пользователю, а какие из них сделать доступными для взаимодействия. Имеется в виду, что при доступном просмотре некоторых данных пользователь может не иметь полномочий на их создание, редактирование или удаление.

* 1. Расчет версионных данных должен производиться на отдельном суперкомпьютере (СК), более мощном, чем сервер, а так как запросов на расчет может быть много, необходимо распределять задачи на расчет между несколькими СК и осуществлять обратное взаимодействие.

Изображение выглядит как текст, карта

Автоматически созданное описание

Применяется балансировщик нагрузки. Пользователь посылает запрос на расчет версии, нажав на кнопку в веб-интерфейсе. При этом происходит запрос к back-end, который делает запись в БД, что версия с данным ID должна быть поставлена на расчет. В БД имеется фоновая задача, проверяющая наличие новых записей в соответствующей таблице. При появлении таковых она создает новое сообщение в очереди сообщений RabbitMQ. Наконец, дело доходит до балансировщика. Тот видит, что есть новый запрос на расчет и при помощи определенного алгоритма, учитывающего работоспособность и загрузку узлов, выбирает один из них, который совершит расчет. С другой стороны, балансировщик в то же время проверяет, нет ли готовых расчетов на узлах, и если есть, записывает результат в БД, которая затем изменяет запись в первоначальной таблице, так что back-end может быть проинформирован о завершении расчет и передать сообщение об этом на front-end.

1. Выводы

Архитектурные проблемы из первой лабораторной работы были проанализированы, и для трех из них в настоящем отчете представлено решение. Для документирования решения было применено схемное и словесное описание. Схемы были созданы с помощью UML, как наиболее известного и простого языка, в то же время предоставляющего мощный функционал.

В сущности, все использованные диаграммы UML так или иначе показывают как структуру системы, так и связи между ее структурными элементами, но частные реализации уделяют больше внимания определенными характеристикам: одни более полно отображаю

т связи, другие – способ взаимодействия и последовательность событий.

2.1Изображение выглядит как снимок экрана

Автоматически созданное описание

2.3Изображение выглядит как снимок экрана

Автоматически созданное описание

Если стрелка раздваивается, это значит, что она сначала происходит обращение к источнику данных, а затем переход к условному блоку или действию.