МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра математических проблем управления и информатики

|  |  |
| --- | --- |
|  | Допущена к защите  Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Смородин "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. |

**Программное средство учёта и контроля рабочего времени сотрудников на базе языка программирования Java и фреймворка ReactJS**

Дипломная работа

Исполнитель

студент группы ПМ-41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пашук А.С.

Научный руководитель

к.т.н.,доцент кафедры МПУиИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Клименко А.В.

Рецензент

доцент кафедры ВМиП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лубочкин А.В.

Гомель 2020

**Содержание**

Оглавление

[Оглавление 2](#_Toc40912595)

[Введение 3](#_Toc40912596)

[1 Сведения об используемых технологиях 4](#_Toc40912597)

[1.1 Общие сведения о языке Java 4](#_Toc40912598)

[1.2 Общие сведения о MySQL 4](#_Toc40912599)

[1.3 Общая информация о разработанной системе 5](#_Toc40912600)

[2 Среда разработки IntelliJ IDEA 6](#_Toc40912601)

[3 Текстовый редактор Atom 7](#_Toc40912602)

[4 Инструмент MySQL Workbench 8](#_Toc40912603)

[5 Технологии используемые при разработке 10](#_Toc40912604)

[5.1 Spring Framework 10](#_Toc40912605)

[5.2 ReactJS 14](#_Toc40912606)

[6 Графическая часть приложения 17](#_Toc40912607)

[6.1 Станица списка сотрудников предприятия 17](#_Toc40912608)

[6.2 Станица добавления нового сотрудника 17](#_Toc40912609)

[6.3 Станица редактирования сотрудника 18](#_Toc40912610)

[6.4 Станица добавления рабочего времени 18](#_Toc40912611)

[6.5 Станица списка рабочего времени 19](#_Toc40912612)

[6.6 Станица отчёта рабочего времени 19](#_Toc40912613)

[Заключение 20](#_Toc40912614)

[Список использованных источников 21](#_Toc40912615)

[Приложение А 22](#_Toc40912616)

[Программный код разработанного приложения ReactJS 22](#_Toc40912617)

[Программный код разработанного приложения Spring Framework 35](#_Toc40912618)

# Введение

Данная работа направлена на разработку веб-приложения для реализации учёта и контроля рабочего времени сотрудников.

Целью данной работы является проектирование веб-сервиса на языке Java с использованием Spring Framework, проектирование базы данных которая будет удовлетворят всем потребностям бизнес логики приложения, разработка клиентского веб-приложения на языке Java Script с использование фреймворка ReactJS.

Для выполнения заданной цели необходимо выполнение нескольких условий:

- знание языка Java;

- знание языка JavaScript;

- знание языка разметки HTML;

- знание языка стилей CSS;

- знание фреймворка ReactJS;

- знание фреймворка Spring Framework;

- знание правил построения REST API;

- знание языка SQL.

Объект исследования - это непосредственно понятие и создание приложения с помощью которого можно реализовать решение поставленной задачи.

Предмет исследования – это разработанное приложение, которое будет описываться в данной работе.

# 1 Сведения об используемых технологиях

## 1.1 Общие сведения о языке Java

Java - [строго](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8_%D1%81%D0%BB%D0%B0%D0%B1%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [типизированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [объектно-ориентированный язык программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанный компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) (в последующем приобретённой компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle)). Разработка ведётся сообществом, организованным через [Java Community Process](https://ru.wikipedia.org/wiki/Java_Community_Process), язык и основные реализующие его технологии распространяются по лицензии [GPL](https://ru.wikipedia.org/wiki/GPL). Права на торговую марку принадлежат корпорации Oracle.

Приложения Java обычно [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в специальный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация [виртуальной Java-машины](https://ru.wikipedia.org/wiki/Java_Virtual_Machine). Дата официального выпуска - 23 мая 1995 года. На 2020 год Java - один из самых популярных языков программирования.

Программы на Java [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [байт-код Java](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4_Java), выполняемый [виртуальной машиной Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) - программой, обрабатывающей байтовый код и передающей инструкции оборудованию как [интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80).

Достоинством подобного способа выполнения программ является полная независимость байт-кода от [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [оборудования](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

## 1.2 Общие сведения о MySQL

MySQL - [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [реляционная система управления базами данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94). Разработку и поддержку MySQL осуществляет корпорация [Oracle](https://ru.wikipedia.org/wiki/Oracle), получившая права на торговую марку вместе с поглощённой [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems), которая ранее приобрела шведскую компанию [MySQL AB](https://ru.wikipedia.org/wiki/MySQL_AB). Продукт распространяется как под [GNU General Public License](https://ru.wikipedia.org/wiki/GNU_General_Public_License), так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм [репликации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)).

Входит в состав серверов [WAMP](https://ru.wikipedia.org/wiki/WAMP), [AppServ](https://ru.wikipedia.org/w/index.php?title=AppServ&action=edit&redlink=1), [LAMP](https://ru.wikipedia.org/wiki/LAMP) и в портативные сборки серверов [Денвер](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)), [XAMPP](https://ru.wikipedia.org/wiki/XAMPP), [VertrigoServ](https://ru.wikipedia.org/w/index.php?title=VertrigoServ&action=edit&redlink=1). Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа [MyISAM](https://ru.wikipedia.org/wiki/MyISAM), поддерживающие полнотекстовый поиск, так и таблицы [InnoDB](https://ru.wikipedia.org/wiki/InnoDB), поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

## 1.3 Общая информация о разработанной системе

Разработано кроссплатформенное клиент-серверное приложение, состоящее из клиентского интерфейса, разработанного с помощью фреймворка ReactJS и серверной части, представляющей из себя REST API построенное на основе фреймворка Spring.

Приложение позволяет вести учёт и статистику рабочего времени сотрудников на предприятии.

# 2 Среда разработки IntelliJ IDEA

IntelliJ IDEA - [интегрированная среда разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для многих языков программирования, в частности [Java](https://ru.wikipedia.org/wiki/Java), [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), [Python](https://ru.wikipedia.org/wiki/Python), разработанная компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains).

Первая версия появилась в январе [2001 года](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3), которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными [свободными](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) инструментами разработчиков, такими как [CVS](https://ru.wikipedia.org/wiki/Concurrent_Versions_System), [Subversion](https://ru.wikipedia.org/wiki/Subversion), [Apache Ant](https://ru.wikipedia.org/wiki/Apache_Ant), [Maven](https://ru.wikipedia.org/wiki/Maven) и [JUnit](https://ru.wikipedia.org/wiki/JUnit). В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке [Ruby](https://ru.wikipedia.org/wiki/Ruby).

Начиная с версии 9.0, среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью [свободной](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) версией, доступной под лицензией [Apache 2.0](https://ru.wikipedia.org/wiki/Apache_License), в ней реализована полная поддержка [Java SE](https://ru.wikipedia.org/wiki/Java_SE), [Kotlin](https://ru.wikipedia.org/wiki/Kotlin), [Groovy](https://ru.wikipedia.org/wiki/Groovy), [Scala](https://ru.wikipedia.org/wiki/Scala_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), а также интеграция с наиболее популярными [системами управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8). В редакции Ultimate Edition, доступной под коммерческой лицензией, реализована поддержка [Java EE](https://ru.wikipedia.org/wiki/Java_EE), [UML](https://ru.wikipedia.org/wiki/UML)-диаграмм, подсчёт [покрытия кода](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0), а также поддержка других систем управления версиями, языков и [фреймворков](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA).

# 3 Текстовый редактор Atom

Atom - бесплатный [текстовый редактор](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80) с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для Linux, macOS, Windows с поддержкой [плагинов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD), написанных на [Node.js](https://ru.wikipedia.org/wiki/Node.js), и встраиваемых под управлением [Git](https://ru.wikipedia.org/wiki/Git). Большинство плагинов имеют статус свободного программного обеспечения, разрабатываются и поддерживаются сообществом.

Atom основан на [Electron](https://ru.wikipedia.org/wiki/Electron) - фреймворке кросс-платформенной разработки с использованием [Chromium](https://ru.wikipedia.org/wiki/Chromium) и [io.js](https://ru.wikipedia.org/wiki/Io.js). Редактор написан на [CoffeeScript](https://ru.wikipedia.org/wiki/CoffeeScript) и [LESS](https://ru.wikipedia.org/wiki/LESS_(%D1%8F%D0%B7%D1%8B%D0%BA_%D1%81%D1%82%D0%B8%D0%BB%D0%B5%D0%B9)). Версия 1.0 была выпущена 25 июня 2015 г. На момент выхода Atom-IDE включает в себя интеллектуальное автодополнение синтаксиса, навигацию по коду, переход к определению функций и классов, поиск ссылок, интерактивные подсказки, выделение синтаксических ошибок, форматирование кода. Изначально релиз поддерживает TypeScript, Flow, JavaScript, Java, C# и PHP и предоставляет средства для глубокого синтаксического анализа кода проектов. В будущем планируется поддержка Rust, Go и Python.

На момент выхода Atom-IDE включает в себя интеллектуальное автодополнение синтаксиса, навигацию по коду, переход к определению функций и классов, поиск ссылок, интерактивные подсказки, выделение синтаксических ошибок, форматирование кода. Изначально релиз поддерживает TypeScript, Flow, JavaScript, Java, C# и PHP и предоставляет средства для глубокого синтаксического анализа кода проектов. В будущем планируется поддержка Rust, Go и Python.

Все компоненты поддержки языков оформлены в виде отдельных серверных обработчиков. Взаимодействие с ними осуществляется с помощью протокола LSP (Language Server Protocol), широко использующегося в продуктах Eclipse, Microsoft и Facebook.

Разработчики советуют использовать Atom Beta 1.21, уже содержащий в себе необходимый мониторинг файлов и управление процессами, для гарантии того, что основные серверные обработчики работают должным образом. Для превращения редактора Atom в среду разработки необходимо установить всего два пакета:

[atom-ide-ui](https://atom.io/packages/atom-ide-ui), делающий доступным пользовательский интерфейс;

необходимый набор поддержки языка программирования, например [ide-typescript](https://atom.io/packages/ide-typescript) для TypeScript, соответственно.

Полный список доступных пакетов можно найти в [блоге разработчиков](https://blog.atom.io/2017/09/12/announcing-atom-ide.html).

# 4 Инструмент MySQL Workbench

MySQL Workbench - инструмент для визуального проектирования [баз данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных [MySQL](https://ru.wikipedia.org/wiki/MySQL). Является преемником [DBDesigner 4](https://ru.wikipedia.org/w/index.php?title=DBDesigner&action=edit&redlink=1) от [FabForce](https://ru.wikipedia.org/w/index.php?title=FabForce&action=edit&redlink=1).

MySQL Workbench предлагается в двух редакциях:

Community Edition - распространяется под свободной лицензией [GNU GPL](https://ru.wikipedia.org/wiki/GNU_GPL);

Standard Edition - доступна по ежегодной оплачиваемой подписке. Эта версия включает в себя дополнительные функции, которые повышают производительность разработчиков и администраторов БД.

Enterprise Edition

Пакет MySQL GUI Tools содержит кросс-платформенное [свободное программное обеспечение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (под лицензией [GNU GPL](https://ru.wikipedia.org/wiki/GNU_GPL)) для администрирования серверов баз данных [MySQL](https://ru.wikipedia.org/wiki/MySQL), а также для создания и манипулирования данными внутри баз данных MySQL. Данный пакет разрабатывался [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems), но позже его разработка была остановлена, и сейчас он доступен только из архивов загрузок на сайте MySQL.

Пакет GUI Tools был полностью заменён MySQL Workbench начиная с бета-версии 5.2. Тем не менее, MySQL Support Team продолжала оказывать поддержку пакета GUI Tools до 30 июня 2010 г. В будущих версиях MySQL Workbench будет добавлен [плагин](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD) для миграции ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) migration plugin), полностью совместимый с MySQL Migration Toolkit, входящий в состав пакета GUI Tools.

Первая версия MySQL Workbench была выпущена в сентябре 2005 года.

MySQL Workbench был первым семейством продуктов, который был доступен в двух вариантах. Чтобы привлечь разработчиков в основную команду разработки, коммерческая стандартная версия программы ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Standard Edition) предлагается поверх свободной версии ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Community Edition), распространяемой под лицензией [GNU GPL](https://ru.wikipedia.org/wiki/GNU_GPL). «Community Edition» является полнофункциональным продуктом, обладающим всеми основными возможностями коммерческого варианта. Являясь основой для всех будущих релизов, он будет получать пользу от всех будущих усилий, прилагаемых для развития продукта. «Standart Edition» расширяет «Community Edition» серией модулей и плагинов, позволяющих оптимизировать рабочий процесс и, тем самым, сэкономить время и избежать ошибок.

Возможности программы:

Позволяет наглядно представить модель базы данных в графическом виде.

Наглядный и функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей.

Reverse Engineering - восстановление структуры таблиц из уже существующей на сервере БД (связи восстанавливаются в [InnoDB](https://ru.wikipedia.org/wiki/InnoDB), при использовании [MyISAM](https://ru.wikipedia.org/wiki/MyISAM) - связи необходимо устанавливать вручную).

Удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы.

Возможность редактирования данных в таблице в визуальном режиме.

# 5 Технологии используемые при разработке

## 5.1 Spring Framework

Spring Framework - универсальный [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) [с открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для [Java](https://ru.wikipedia.org/wiki/Java)-платформы. Также существует [форк](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BA) для платформы [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), названный Spring.NET.

Первая версия была написана [Родом Джонсоном](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B6%D0%BE%D0%BD%D1%81%D0%BE%D0%BD,_%D0%A0%D0%BE%D0%B4&action=edit&redlink=1), который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» ([Wrox Press](https://ru.wikipedia.org/wiki/Wrox_Press), [октябрь](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%82%D1%8F%D0%B1%D1%80%D1%8C) [2002 года](https://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4)).

Фреймворк был впервые выпущен под лицензией [Apache 2.0 license](https://ru.wikipedia.org/wiki/Apache_License) в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте [2004](https://ru.wikipedia.org/wiki/2004). Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.2.4.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели [Enterprise JavaBeans](https://ru.wikipedia.org/wiki/EJB). Spring предоставляет бо́льшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) на [Java Enterprise платформе](https://ru.wikipedia.org/wiki/J2EE). По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе [Java](https://ru.wikipedia.org/wiki/Java). Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. Spring не всецело связан с платформой [Java Enterprise](https://ru.wikipedia.org/wiki/J2EE), несмотря на его масштабную интеграцию с ней, что является важной причиной его популярности.

Spring, вероятно, наиболее известен как источник расширений (features), нужных для эффективной разработки сложных бизнес-приложений вне тяжеловесных программных моделей, которые исторически были доминирующими в промышленности. Ещё одно его достоинство в том, что он ввел ранее неиспользуемые функциональные возможности в сегодняшние господствующие методы разработки, даже вне платформы Java.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что Spring реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java.



**Рисунок 1 – Spring IoC**

Центральной частью Spring является контейнер [Inversion of Control](https://ru.wikipedia.org/wiki/Inversion_of_Control), который предоставляет средства конфигурирования и управления объектами Java с помощью [рефлексии](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%BB%D0%B5%D0%BA%D1%81%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Контейнер отвечает за управление жизненным циклом объекта: создание объектов, вызов методов инициализации и конфигурирование объектов путём связывания их между собой.

Объекты, создаваемые контейнером, также называются управляемыми объектами (beans). Обычно, конфигурирование контейнера, осуществляется путём внедрения аннотаций (начиная с [5 версии J2SE](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9_Java_SE#J2SE_5.0_(30_%D1%81%D0%B5%D0%BD%D1%82%D1%8F%D0%B1%D1%80%D1%8F_2004))), но также, есть возможность, по старинке, загрузить XML-файлы, содержащие определение bean’ов и предоставляющие информацию, необходимую для создания bean’ов.



**Рисунок 2 – Spring MVC**

Spring имеет собственную [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller)-платформу веб-приложений, которая не была первоначально запланирована. Разработчики Spring решили написать её как реакцию на то, что они восприняли как неудачность конструкции (тогда) популярного [Apache Struts](https://ru.wikipedia.org/wiki/Apache_Struts), а также других доступных веб-фреймворков. В частности, по их мнению, было недостаточным разделение между слоями представления и обработки запросов, а также между слоем обработки запросов и моделью.

Класс [DispatcherServlet](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/servlet/DispatcherServlet.html) является основным контроллером фреймворка и отвечает за делегирование управления различным интерфейсам, на всех этапах выполнения HTTP-запроса. Об этих интерфейсах следует сказать более подробно.

Как и Struts, Spring MVC является фреймворком, ориентированным на запросы. В нем определены [стратегические](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) интерфейсы для всех функций современной запросно-ориентированной системы. Цель каждого интерфейса — быть простым и ясным, чтобы пользователям было легко его заново имплементировать, если они того пожелают. MVC прокладывает путь к более чистому front-end-коду. Все интерфейсы тесно связаны с [Servlet API](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82_(Java)). Эта связь рассматривается некоторыми как неспособность разработчиков Spring предложить для веб-приложений абстракцию более высокого уровня. Однако эта связь оставляет особенности Servlet API доступными для разработчиков, облегчая все же работу с ним. Наиболее важные интерфейсы, определенные Spring MVC, перечислены ниже:

[HandlerMapping](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/portlet/HandlerMapping.html): выбор класса и его метода, которые должны обработать данный входящий запрос на основе любого внутреннего или внешнего для этого запроса атрибута или состояния.

[HandlerAdapter](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/portlet/HandlerAdapter.html): вызов и выполнение выбранного метода обработки входящего запроса.

[Controller](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/servlet/mvc/Controller.html): включен между Моделью (Model) и Представлением (View). Управляет процессом преобразования входящих запросов в адекватные ответы. Действует как ворота, направляющие всю поступающую информацию. Переключает поток информации из модели в представление и обратно.

[View](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/servlet/View.html): ответственно за возвращение ответа клиенту в виде текстов и изображений. Некоторые запросы могут идти прямо во View, не заходя в Model; другие проходят через все три слоя.

[ViewResolver](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/servlet/ViewResolver.html): выбор, какое именно View должно быть показано клиенту.

[HandlerInterceptor](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/portlet/HandlerInterceptor.html): перехват входящих запросов. Сопоставим, но не эквивалентен сервлет-фильтрам (использование не является обязательным и не контролируется DispatcherServlet-ом).

[LocaleResolver](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/servlet/LocaleResolver.html): получение и, возможно, сохранение локальных настроек (язык, страна, часовой пояс) пользователя.

[MultipartResolver](http://static.springsource.org/spring/docs/3.2.x/javadoc-api/org/springframework/web/multipart/MultipartResolver.html): обеспечивает Upload — загрузку на сервер локальных файлов клиента.

[Spring MVC](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html) предоставляет разработчику следующие возможности:

Ясное и прозрачное разделение между слоями в MVC и запросах.

Стратегия интерфейсов — каждый интерфейс делает только свою часть работы.

Интерфейс всегда может быть заменен альтернативной реализацией.

Интерфейсы тесно связаны с Servlet API.

Высокий уровень абстракции для веб-приложений.

В веб-приложениях можно использовать различные части Spring, а не только Spring MVC.

## 5.2 ReactJS

React (иногда React.js или ReactJS) - [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)-[библиотека](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript) с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для разработки [пользовательских интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F).

React разрабатывается и поддерживается [Facebook](https://ru.wikipedia.org/wiki/Facebook), [Instagram](https://ru.wikipedia.org/wiki/Instagram) и сообществом отдельных разработчиков и корпораций.

React может использоваться для разработки [одностраничных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как [Redux](https://ru.wikipedia.org/wiki/Redux) и GraphQL.

React был создан Джорданом Валке, разработчиком программного обеспечения из Facebook. На него оказал влияние [XHP](https://ru.wikipedia.org/w/index.php?title=XHP&action=edit&redlink=1) — компонентный [HTML](https://ru.wikipedia.org/wiki/HTML)-фреймворк для [PHP](https://ru.wikipedia.org/wiki/PHP). Впервые React использовался в новостной ленте Facebook в 2011 году и позже в ленте Instagram в [2012 году](https://ru.wikipedia.org/wiki/2012_%D0%B3%D0%BE%D0%B4). Исходный код React открыт в мае 2013 года на конференции «JSConf US».

React Native анонсирован на конференции Facebook «React.js Conf» в [феврале](https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D0%B2%D1%80%D0%B0%D0%BB%D1%8C) [2015 года](https://ru.wikipedia.org/wiki/2015_%D0%B3%D0%BE%D0%B4), а [исходный код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) открыт в [марте](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%82) [2015 года](https://ru.wikipedia.org/wiki/2015_%D0%B3%D0%BE%D0%B4). Он позволяет разрабатывать [нативные](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F) [Android](https://ru.wikipedia.org/wiki/Android)-, [iOS](https://ru.wikipedia.org/wiki/IOS)- и [UWP](https://ru.wikipedia.org/wiki/Universal_Windows_Platform)-приложения с использованием React.

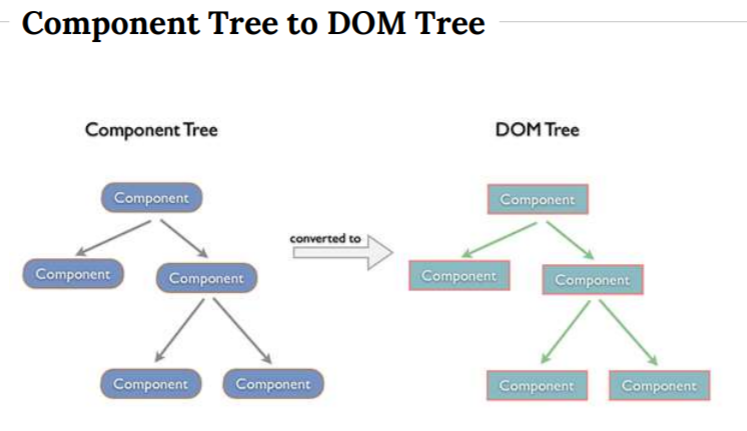
[18 апреля](https://ru.wikipedia.org/wiki/18_%D0%B0%D0%BF%D1%80%D0%B5%D0%BB%D1%8F) [2017 года](https://ru.wikipedia.org/wiki/2017_%D0%B3%D0%BE%D0%B4) Facebook анонсировал [React Fiber](https://ru.wikipedia.org/w/index.php?title=React_Fiber&action=edit&redlink=1), переписанную и оптимизированную версию React. React Fiber станет основой разработки всех будущих функций и улучшений.

Свойства передаются от родительских компонентов к дочерним. Компоненты получают свойства как множество [неизменяемых](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) immutable) значений, поэтому компонент не может напрямую изменять свойства, но может вызывать изменения через [callback-функции](https://ru.wikipedia.org/wiki/Callback_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Такой механизм называют «свойства вниз, события наверх».

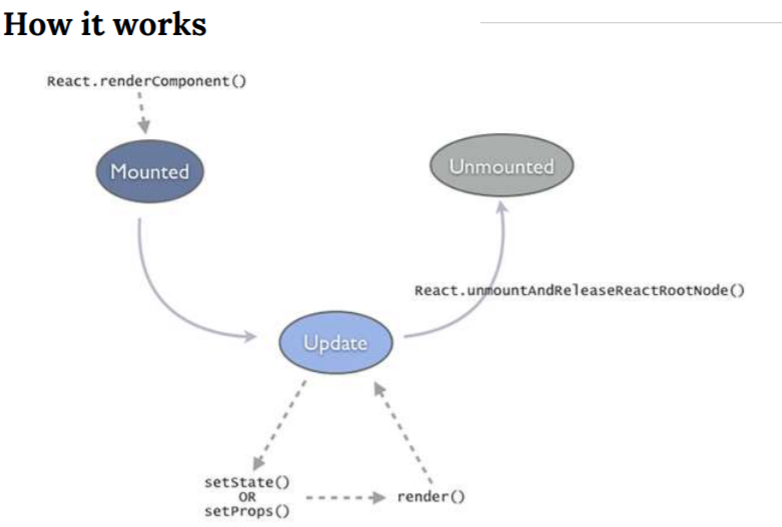
React использует виртуальный [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) virtual DOM). React создаёт [кэш](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88)-структуру в памяти, что позволяет вычислять разницу между предыдущим и текущим состояниями интерфейса для оптимального обновления DOM браузера. Таким образом программист может работать со страницей, считая, что она обновляется вся, но библиотека самостоятельно решает, какие компоненты страницы необходимо обновить.

JavaScript XML (JSX) — это расширение синтаксиса JavaScript, которое позволяет использовать HTML-подобный синтаксис для описания структуры интерфейса. Как правило, компоненты написаны с использованием JSX, но также есть возможность использования обычного [JavaScript](https://ru.wikipedia.org/wiki/JavaScript). JSX напоминает другой язык, созданный в компании Фейсбук для расширения PHP, XHP .

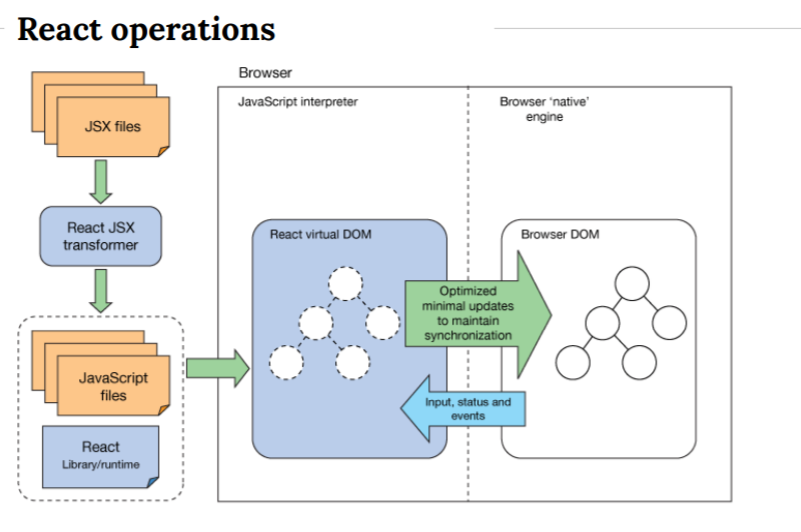
React используется не только для отрисовки HTML в браузере. Например, Facebook имеет динамические графики, которые отрисовываются в теги <canvas>. Netflix и PayPal используют изоморфные загрузки для отрисовки идентичного HTML на сервере и клиенте.



**Рисунок 3 – ReactJS Component Tree**



**Рисунок 4 – ReactJS render workflow**

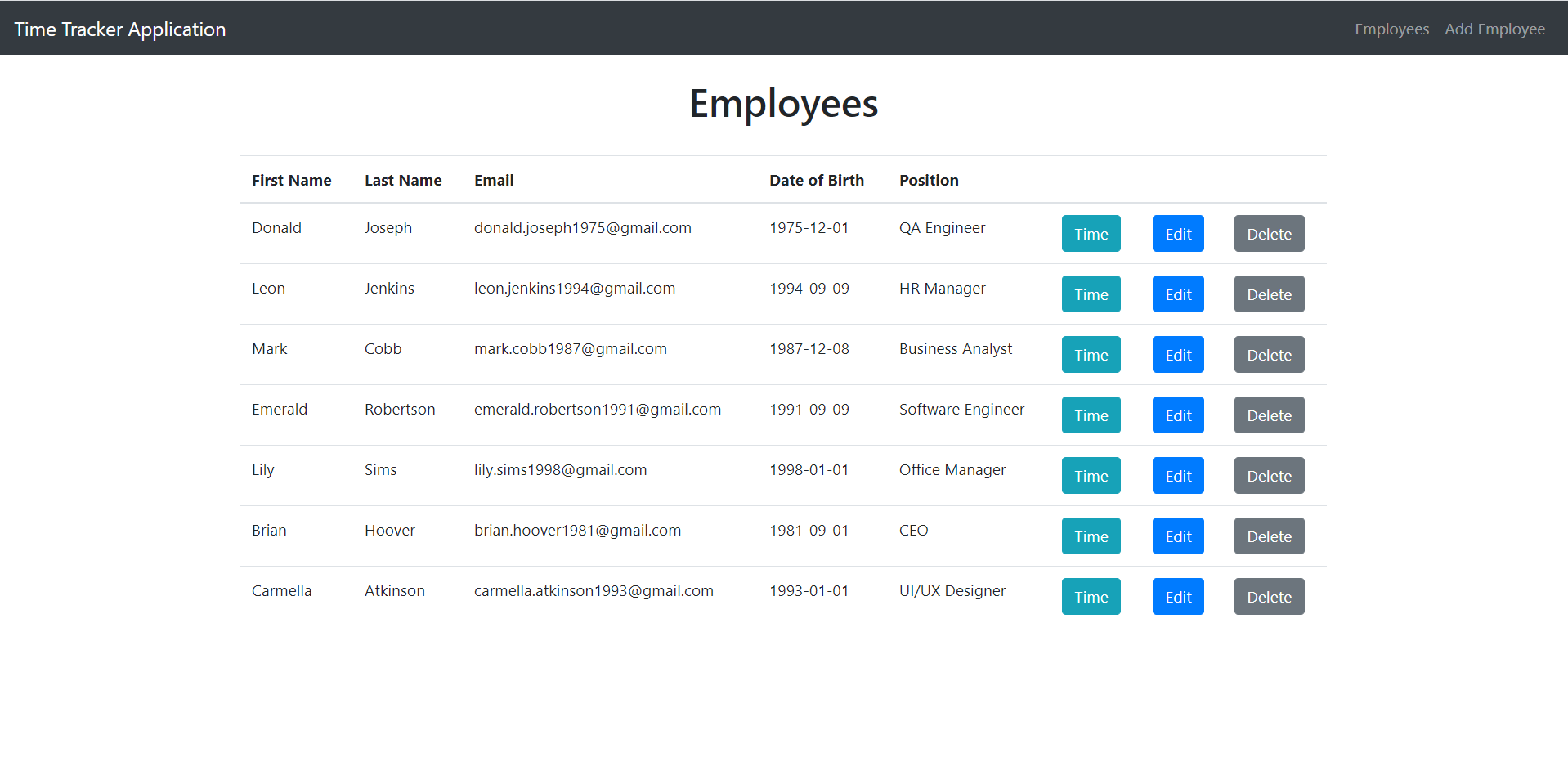


**Рисунок 5 – ReactJS operations**

# 6 Графическая часть приложения

## 6.1 Страница списка сотрудников предприятия

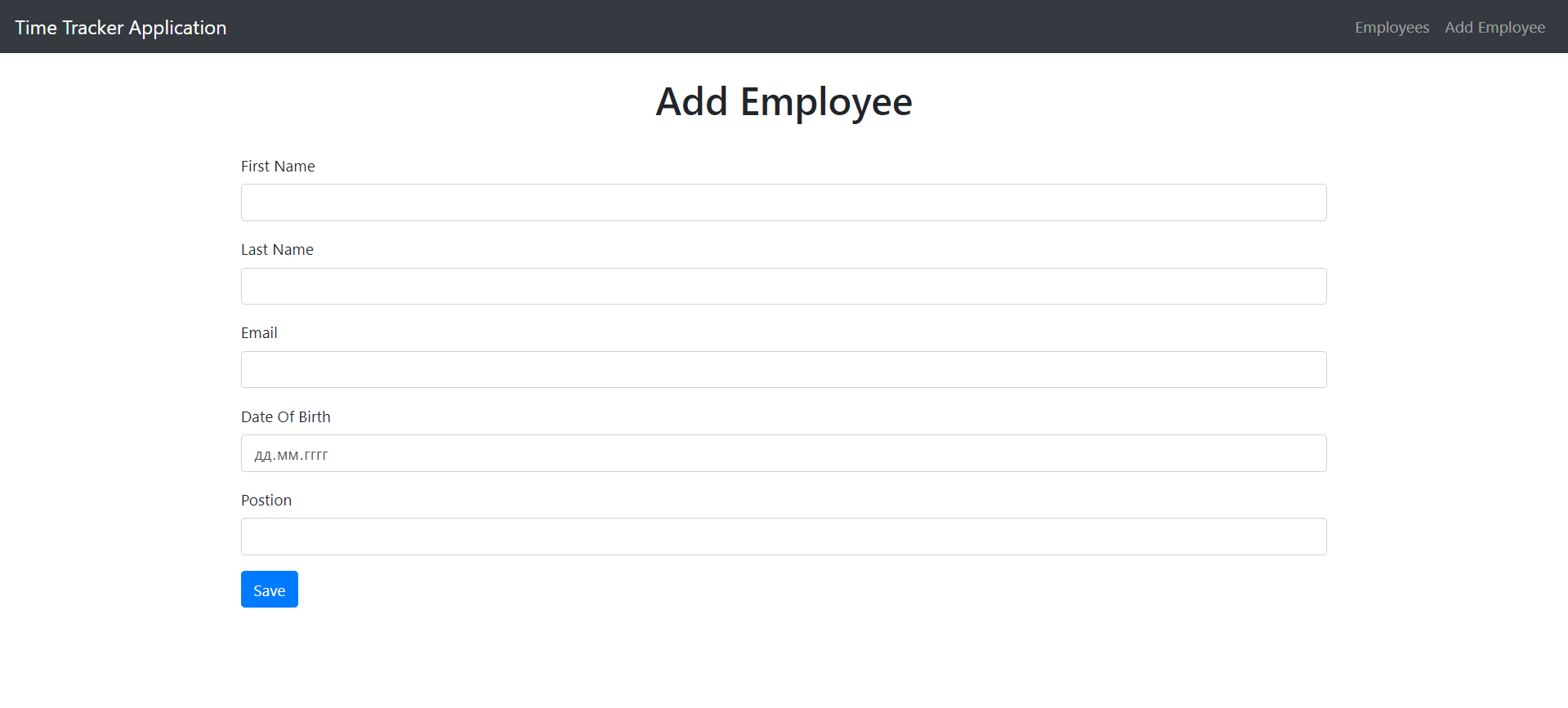
На рисунке 6 показана страница списка сотрудников предприятия



**Рисунок 6 – Страница списка сотрудников предприятия**

## 6.2 Страница добавления нового сотрудника

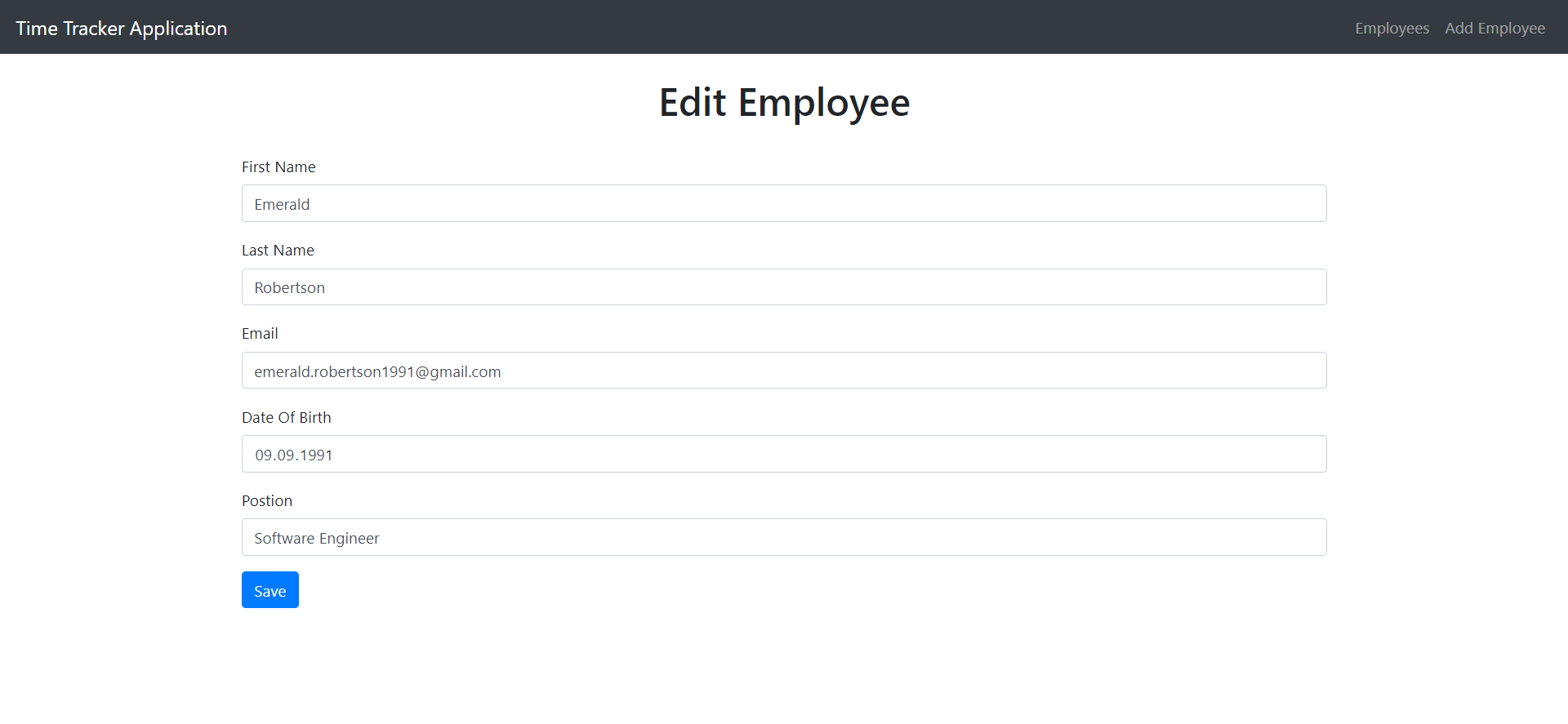
На рисунке 7 показана страница добавления сотрудника



**Рисунок 7 – Страница добавления сотрудника**

## 6.3 Страница редактирования сотрудника

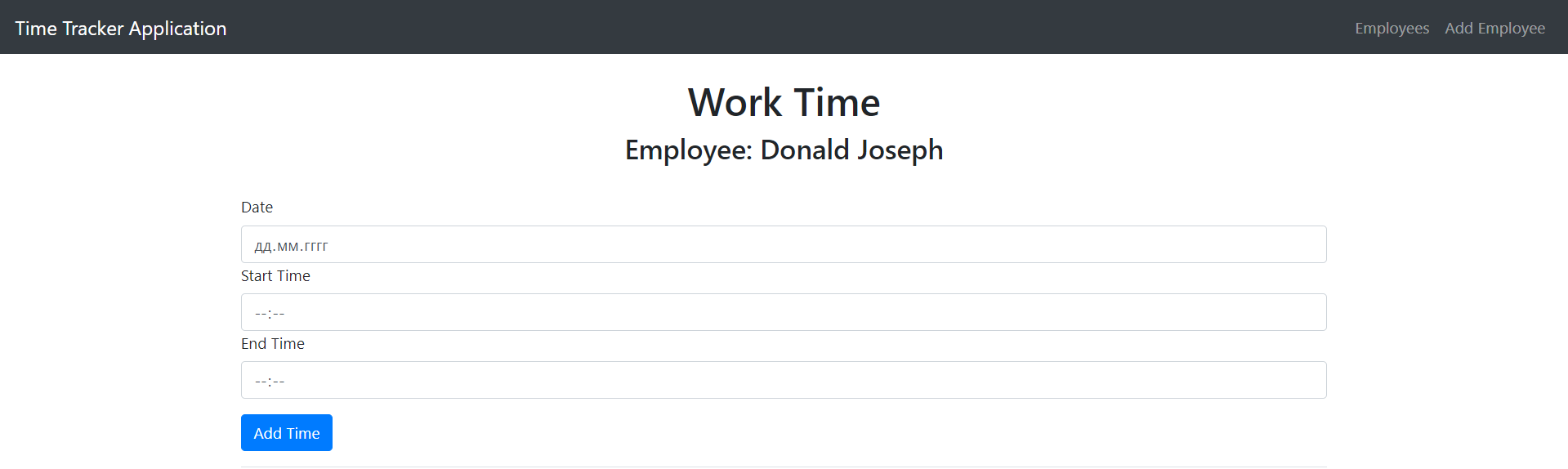
На рисунке 8 показана страница редактирования сотрудника



**Рисунок 8 – Страница редактирования сотрудника**

## 6.4 Страница добавления рабочего времени

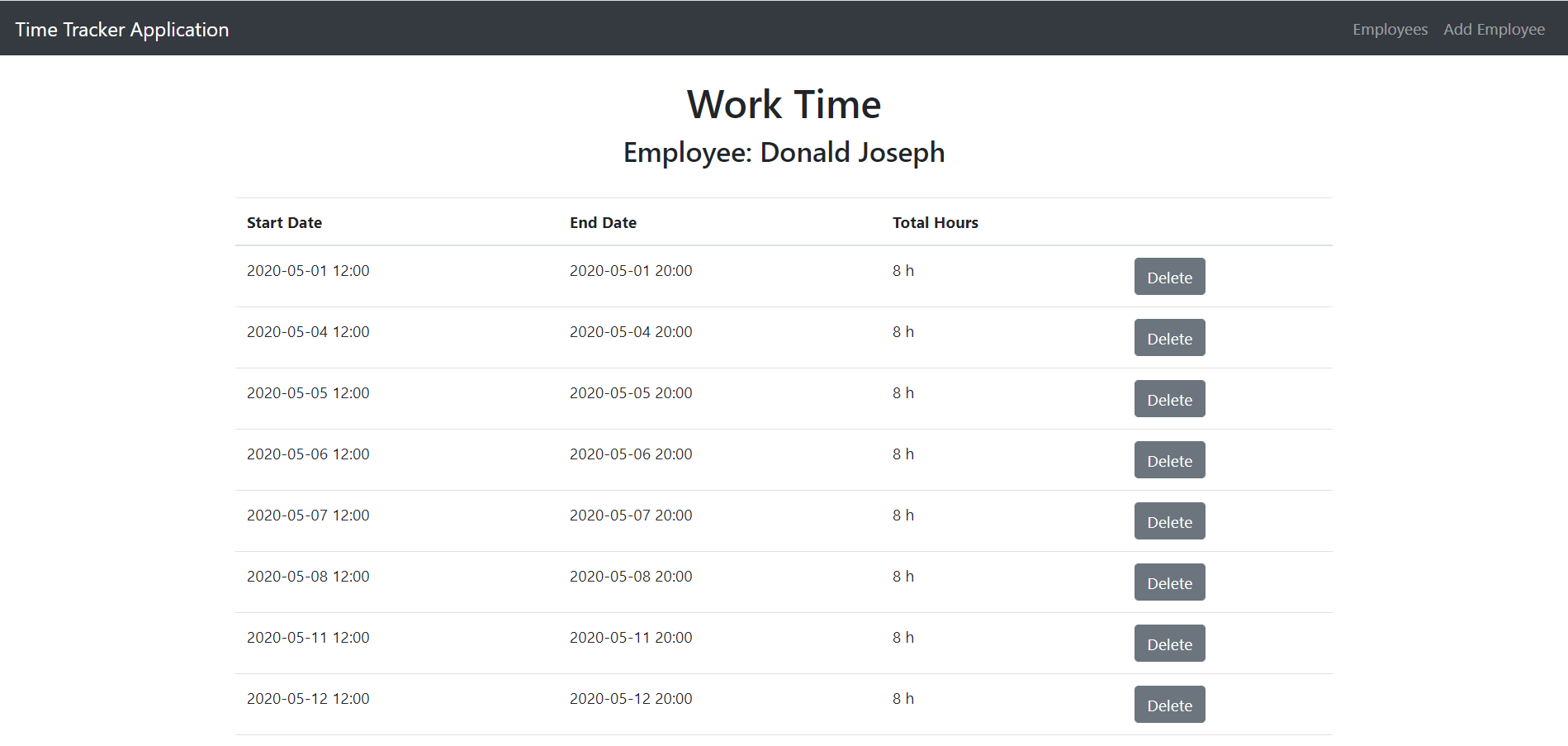
На рисунке 9 показана страница добавления рабочего времени



**Рисунок 9 – Страница добавления рабочего времени**

## 6.5 Страница списка рабочего времени

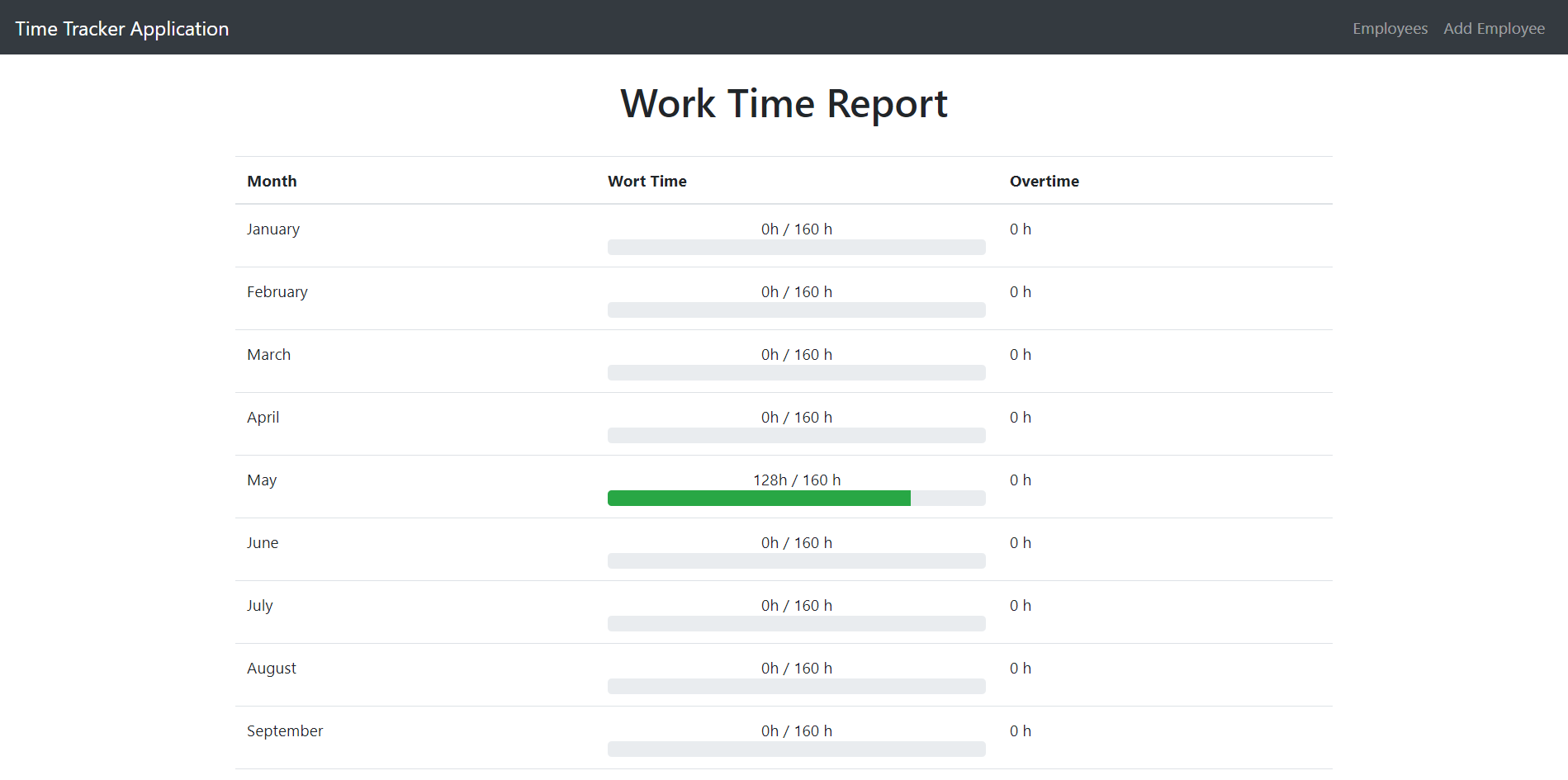
На рисунке 10 показана страница списка сотрудников предприятия



**Рисунок 10 – Страница списка рабочего времени**

## 6.6 Страница отчёта рабочего времени

На рисунке 11 показана страница отчёта рабочего времени



**Рисунок 11 – Страница отчёта рабочего времени**

# Заключение

В данной работе был изучен классический по современным меркам стек технологий для веб-приложений. В процессе разработки было выявлено множество преимуществ использования разделённого стека. В рамках данной работы был разработан REST сервис и клиентское веб приложение, но благодаря данному подходу остаётся возможность разрабатывать дополнительные клиентские приложения на различных платформах (например: Android, iOS), переиспользуя серверную часть. Данный подход невозможно реализовать, используя устаревшие монолитные архитектурные решения такие как JSP, JSF и т.д.

# Список использованных источников

1. Загребаев, А. Методы математического программирования в задачах оптимизации сложных технических систем[Текст]/А.Загребаев – М.:МИФИ 2007. — 332с.

2. Васильев,Ф. Численные методы решения экстремальных задач. Учебное пособие[Текст]/Ф.Васильев - М.: Наука, 1958. — 549 с.

3. Габасов,Р. Кириллова Ф.М. Методы оптимизации[Текст]/Р.Габасов - Мн.: Изд. бел. гос. ун., 1981. — 350с.

4. Руководство разработчика Borland C++ Builder 6[Текст]/ Джаррод Холингворт[и др.]. — М.: «Вильямс», 2004. — 976 c.

Приложение А

## Программный код разработанного приложения ReactJS

import React, { Component } from 'react';

import NavigationBar from './NavigationBar';

import { Table, Container, Input, Button, Label, FormGroup, Form} from 'reactstrap';

class AddEmployee extends Component {

    constructor(props){

        super(props)

        this.state = {

            firstName: '',

            lastName: '',

            email: '',

            dateOfBirth: new Date(),

            position: ''

        }

         this.handleSubmit = this.handleSubmit.bind(this);

         this.handleChange = this.handleChange.bind(this);

      }

    handleChange(event) {

        const name = event.target.name;

        const value = event.target.value;

        this.setState({

          [name]: value

        })

    }

    async handleSubmit(event){

        await fetch(`http://localhost:8080/api/employee`, {

          method : 'POST',

          headers : {

            'Accept': 'application/json',

            'Content-Type': 'application/json'

          },

          body : JSON.stringify(this.state),

        }).then(() => {

            alert("Employee has been successfully created.");

          });

        event.preventDefault();

      }

    render() {

        return(

            <div>

                <NavigationBar />

                <Container>

                    <br></br>

                    <h1 style={{textAlign: 'center'}}>Add Employee</h1>

                    <br></br>

                    <Form onSubmit={this.handleSubmit}>

                        <FormGroup>

                            <Label for="firstName">First Name</Label>

                            <Input type="text" name="firstName" id="firstName" required

                                onChange={this.handleChange}/>

                        </FormGroup>

                        <FormGroup>

                            <Label for="lastName">Last Name</Label>

                            <Input type="text" name="lastName" id="lastName" required

                                onChange={this.handleChange}/>

                        </FormGroup>

                        <FormGroup>

                            <Label for="email">Email</Label>

                            <Input type="email" name="email" id="email" required

                                onChange={this.handleChange}/>

                        </FormGroup>

                        <FormGroup>

                            <Label for="dateOfBirth">Date Of Birth</Label>

                            <Input type="date" name="dateOfBirth" id="dateOfBirth" required

                                onChange={this.handleChange}/>

                        </FormGroup>

                        <FormGroup>

                            <Label for="position">Postion</Label>

                            <Input type="text" name="position" id="position" required

                                onChange={this.handleChange}/>

                        </FormGroup>

                        <FormGroup>

                            <Button color="primary" type="submit">Save</Button>

                        </FormGroup>

                    </Form>

                </Container>

            </div>

        );

    }

}

export default AddEmployee;

import React, { Component } from 'react';

import { Route, BrowserRouter as Router, Switch } from 'react-router-dom'

import Employees from './Employees';

import Home from './Home';

import AddEmployee from './AddEmployee';

import EditEmployee from './EditEmployee';

import WorkTime from './WorkTime';

import WorkTimeReport from './WorkTimeReport';

class App extends Component {

    state = {  }

    render() {

        return (

            <Router>

                <Switch>

                    <Route path='/' exact={true} component={Home}/>

                    <Route path='/employees' exact={true} component={Employees}/>

                    <Route path='/employee/add' exact={true} component={AddEmployee}/>

                    <Route path='/employee/edit/:id' exact={true} component={EditEmployee}/>

                    <Route path='/employee/time/:id' exact={true} component={WorkTime}/>

                    <Route path='/employee/times/year/:year/:id' exact={true} component={WorkTimeReport}/>

                </Switch>

             </Router>

        );

    }

}

export default App;

import React, { Component } from 'react';

import NavigationBar from './NavigationBar';

import { Container, Input, Button, Label, FormGroup, Form } from 'reactstrap';

class EditEmployee extends Component {

    constructor(props) {

        super(props)

        this.state = {

            id: 0,

            firstName: '',

            lastName: '',

            email: '',

            dateOfBirth: new Date(),

            position: ''

        }

        this.handleSubmit = this.handleSubmit.bind(this);

        this.handleChange = this.handleChange.bind(this);

    }

    async componentDidMount() {

        const response = await fetch('http://localhost:8080/api/employee/' + this.props.match.params.id);

        const json = await response.json();

        this.setState(

            {

                id: json.id,

                firstName: json.firstName,

                lastName: json.lastName,

                email: json.email,

                dateOfBirth: json.dateOfBirth.substring(0,10),

                position: json.position

            }

        );

    }

    handleChange(event) {

        const name = event.target.name;

        const value = event.target.value;

        this.setState({

          [name]: value

        })

    }

    async handleSubmit(event) {

        await fetch(`http://localhost:8080/api/employee/edit`, {

          method : 'POST',

          headers : {

            'Accept': 'application/json',

            'Content-Type': 'application/json'

          },

          body : JSON.stringify(this.state),

        }).then(() => {

            alert("Employee has been successfully edited.");

          });

        event.preventDefault();

    }

    render() {

        return(

            <div>

                <NavigationBar />

                <Container>

                    <br></br>

                    <h1 style={{textAlign: 'center'}}>Edit Employee</h1>

                    <br></br>

                    <Form onSubmit={this.handleSubmit}>

                        <FormGroup>

                            <Label for="firstName">First Name</Label>

                            <Input type="text" name="firstName" id="firstName" required

                                onChange={this.handleChange} value={this.state.firstName} />

                        </FormGroup>

                        <FormGroup>

                            <Label for="lastName">Last Name</Label>

                            <Input type="text" name="lastName" id="lastName" required

                                onChange={this.handleChange} value={this.state.lastName} />

                        </FormGroup>

                        <FormGroup>

                            <Label for="email">Email</Label>

                            <Input type="email" name="email" id="email" required

                                onChange={this.handleChange} value={this.state.email} />

                        </FormGroup>

                        <FormGroup>

                            <Label for="dateOfBirth">Date Of Birth</Label>

                            <Input type="date" name="dateOfBirth" id="dateOfBirth" required

                                onChange={this.handleChange} value={this.state.dateOfBirth} />

                        </FormGroup>

                        <FormGroup>

                            <Label for="position">Postion</Label>

                            <Input type="text" name="position" id="position" required

                                onChange={this.handleChange} value={this.state.position} />

                        </FormGroup>

                        <FormGroup>

                            <Button color="primary" type="submit">Save</Button>

                        </FormGroup>

                    </Form>

                </Container>

            </div>

        );

    }

}

export default EditEmployee;

import React, { Component } from 'react';

import NavigationBar from './NavigationBar';

import { Table, Container, Button } from 'reactstrap';

import { Link } from 'react-router-dom';

class Employees extends Component {

    state = {

        employees : []

    }

    async componentDidMount() {

        const response = await fetch('http://localhost:8080/api/employees');

        const json = await response.json();

        this.setState(

           {employees : json}

        );

    }

    async delete(id) {

        await fetch(`http://localhost:8080/api/employee/delete/${id}` , {

            method: 'DELETE'

          }).then(() => {

            this.setState({

                employees: this.state.employees.filter(employee => employee.id !== id)

              });

          });

    }

    render() {

        const employees = this.state.employees;

        console.log(employees);

        return(

            <div>

                <NavigationBar />

                <Container>

                    <br></br>

                    <h1 style={{textAlign: 'center'}}>Employees</h1>

                    <br></br>

                    <Table hover>

                        <thead>

                            <tr>

                                <th>First Name</th>

                                <th>Last Name</th>

                                <th>Email</th>

                                <th>Date of Birth</th>

                                <th>Position</th>

                                <th></th>

                                <th></th>

                                <th></th>

                            </tr>

                        </thead>

                        <tbody>

                            {

                                employees.map( employee =>

                                    <tr>

                                        <td>{employee.firstName}</td>

                                        <td>{employee.lastName}</td>

                                        <td>{employee.email}</td>

                                        <td>{employee.dateOfBirth.substring(0,10)}</td>

                                        <td>{employee.position}</td>

                                        <td><Link to={{ pathname: `/employee/time/${employee.id}`}}><Button color="info">Time</Button></Link></td>

                                        <td><Link to={{ pathname: `/employee/edit/${employee.id}`}}><Button color="primary">Edit</Button></Link></td>

                                        <td><Button color="secondary" onClick={() => {if (window.confirm('Are you sure you wish to delete this employee?')) this.delete(employee.id)}}>Delete</Button></td>

                                    </tr>

                                )

                            }

                        </tbody>

                    </Table>

                </Container>

            </div>

        );

    }

}

export default Employees;

import React, { Component } from 'react';

import NavigationBar from './NavigationBar';

class Home extends Component {

    state = {}

    render() {

        return(

            <div>

                <NavigationBar />

            </div>

        );

    }

}

export default Home;

import React, { Component } from 'react';

import {Nav, NavItem, NavLink, Navbar, NavbarBrand} from 'reactstrap';

class NavigationBar extends Component {

    state = {}

    render() {

        return (

            <div>

              <Navbar color="dark" dark expand="md">

                <NavbarBrand href="/">Time Tracker Application</NavbarBrand>

                  <Nav className="ml-auto" navbar>

                    <NavItem>

                      <NavLink href="/employees/">Employees</NavLink>

                    </NavItem>

                    <NavItem>

                      <NavLink href="/employee/add">Add Employee</NavLink>

                    </NavItem>

                  </Nav>

              </Navbar>

            </div>

        );

    }

}

export default NavigationBar;

import React, { Component, useState  } from 'react';

import NavigationBar from './NavigationBar';

import { Container, Input, Button, Label, FormGroup, Form, Table } from 'reactstrap';

import { Link } from 'react-router-dom';

import Moment from 'moment'

class WorkTime extends Component {

    constructor(props) {

        super(props)

        this.state = {

            id: 0,

            firstName: '',

            lastName: '',

            email: '',

            dateOfBirth: new Date(),

            position: '',

            timePeriods: [],

            startDate: '',

            startTime: '',

            endTime: ''

        }

        this.handleSubmit = this.handleSubmit.bind(this);

        this.handleChange = this.handleChange.bind(this);

    }

    async componentDidMount() {

        const response = await fetch('http://localhost:8080/api/employee/' + this.props.match.params.id);

        const json = await response.json();

        this.setState({

            id: json.id,

            firstName: json.firstName,

            lastName: json.lastName,

            email: json.email,

            dateOfBirth: json.dateOfBirth.substring(0,10),

            position: json.position,

            timePeriods: json.timePeriods

        });

    }

    handleChange(event) {

        const name = event.target.name;

        const value = event.target.value;

        this.setState({

          [name]: value

        });

    }

    async handleSubmit(event) {

        let response = await fetch(`http://localhost:8080/api/addTime/` + this.state.id, {

          method : 'POST',

          headers : {

            'Accept': 'application/json',

            'Content-Type': 'application/json'

          },

          body : JSON.stringify({

            startDate: this.state.startDate + 'T' + this.state.startTime + ':00.000+0000',

            endDate: this.state.startDate + 'T' + this.state.endTime + ':00.000+0000',

            employee: {

                id: this.state.id,

                firstName: this.state.firstName,

                lastName: this.state.lastName,

                email: this.state.email,

                dateOfBirth: this.state.dateOfBirth,

                position: this.state.position

            }

          }),

        });

        if (!response.ok) {

            alert("End Time must be greater than Start Time!");

        } else {

            alert("Time Period has been added.")

        }

        event.preventDefault();

    }

    async delete(id) {

        await fetch(`http://localhost:8080/api/time/delete/${id}` , {

            method: 'DELETE'

        }).then(() => {

            this.setState({

                timePeriods: this.state.timePeriods.filter(timePeriod => timePeriod.id !== id)

            });

        });

    }

    render() {

        Moment.locale('en');

        return(

            <div>

                <NavigationBar />

                <Container>

                    <br></br>

                    <h1 style={{textAlign: 'center'}}>Work Time</h1>

                    <h3 style={{textAlign: 'center'}}>Employee: {this.state.firstName} {this.state.lastName}</h3>

                    <br></br>

                    <div>

                        <Form onSubmit={this.handleSubmit}>

                            <FormGroup>

                                <Label for="startDate">Date</Label>

                                <Input type="date" name="startDate" id="startDate" required onChange={this.handleChange}  />

                                <Label>Start Time</Label>

                                <Input type="time" name="startTime" id="startTime" required onChange={this.handleChange}  />

                                <Label >End Time</Label>

                                <Input type="time" name="endTime" id="endTime" required onChange={this.handleChange} />

                            </FormGroup>

                            <FormGroup>

                                <Button color="primary" type="submit">Add Time</Button>{' '}

                            </FormGroup>

                        </Form>

                    </div>

                    <div>

                        <Table hover>

                            <thead>

                                <tr>

                                    <th>Start Date</th>

                                    <th>End Date</th>

                                    <th>Total Hours</th>

                                    <th></th>

                                </tr>

                            </thead>

                            <tbody>

                                {

                                    this.state.timePeriods.map( timePeriod =>

                                        <tr>

                                            <td>{Moment(timePeriod.startDate).zone(0).format('YYYY-MM-DD HH:mm')}</td>

                                            <td>{Moment(timePeriod.endDate).zone(0).format('YYYY-MM-DD HH:mm')}</td>

                                            <td>{Moment(timePeriod.endDate).diff(timePeriod.startDate, 'hours')} h</td>

                                            <td><Button color="secondary" onClick={() => {if (window.confirm('Are you sure you wish to delete this time period?')) this.delete(timePeriod.id)}}>Delete</Button></td>

                                        </tr>

                                    )

                                }

                            </tbody>

                        </Table>

                    </div>

                    <div>

                        <Link to={{ pathname: `/employee/times/year/2020/${this.state.id}`}}><Button color="info">See Year Report</Button></Link>

                    </div>

                    <br></br>

                </Container>

            </div>

        );

    }

}

export default WorkTime;

import React, { Component, useState  } from 'react';

import NavigationBar from './NavigationBar';

import { Container, Input, Button, Label, FormGroup, Form, Table, Progress } from 'reactstrap';

class WorkTimeReport extends Component {

    constructor(props) {

        super(props)

        this.state = {

            items: []

        }

    }

    async componentDidMount() {

        const response = await fetch('http://localhost:8080/api/times/year/' + this.props.match.params.year + '/' + this.props.match.params.id);

        const json = await response.json();

        this.setState(

            {items : json}

         );

    }

    render() {

        return(

            <div>

                <NavigationBar />

                <Container>

                    <br></br>

                    <h1 style={{textAlign: 'center'}}>Work Time Report</h1>

                    <br></br>

                    <Table hover>

                        <thead>

                            <tr>

                                <th>Month</th>

                                <th>Wort Time</th>

                                <th>Overtime</th>

                            </tr>

                        </thead>

                        <tbody>

                            {

                                this.state.items.map( item =>

                                    <tr>

                                        <td>{item.monthName}</td>

                                        <td>

                                            <div className="text-center">{item.hours}h / 160 h</div>

                                            <Progress color="success" value={item.percent}></Progress>

                                        </td>

                                        <td>{((item.hours - 160) > 0) ? item.hours - 160 : '0'} h</td>

                                    </tr>

                                )

                            }

                        </tbody>

                    </Table>

                </Container>

            </div>

        );

    }

}

export default WorkTimeReport;

## Программный код разработанного приложения Spring Framework

package com.example.TimeTracker.api;  
  
import com.example.TimeTracker.model.Employee;  
import com.example.TimeTracker.service.EmployeeService;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/api")  
@CrossOrigin(origins = "http://localhost:3000")  
public class EmployeeController {  
  
 private EmployeeService employeeService;  
  
 public EmployeeController(EmployeeService employeeService) {  
 this.employeeService = employeeService;  
 }  
  
 @GetMapping("/employees")  
 List<Employee> employees() {  
 return this.employeeService.getAllEmployees();  
 }  
  
 @GetMapping("/employee/{id}")  
 ResponseEntity<?> getEmployee(@PathVariable Long id){  
 Optional<Employee> category = employeeService.getEmployeeById(id);  
 return category.map(response -> ResponseEntity.ok().body(response))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
  
 }  
  
 @PostMapping("/employee/edit")  
 ResponseEntity<?> editEmployee(@RequestBody Employee employee) throws URISyntaxException {  
 Employee result = employeeService.updateEmployee(employee);  
 return ResponseEntity.ok().build();  
 }  
  
 @PostMapping("/employee")  
 ResponseEntity<?> createEmployee(@RequestBody Employee employee) throws URISyntaxException {  
 Employee result = employeeService.createEmployee(employee);  
 return ResponseEntity.created(new URI("/api/employee" + result.getId())).body(result);  
  
 }  
  
 @DeleteMapping("/employee/delete/{id}")  
 ResponseEntity<?> deleteEmployee(@PathVariable Long id) {  
 employeeService.deleteEmployeeById(id);  
 return ResponseEntity.ok().build();  
 }  
  
}

package com.example.TimeTracker.api;  
  
import com.example.TimeTracker.dto.MonthWorkTimeDTO;  
import com.example.TimeTracker.model.Employee;  
import com.example.TimeTracker.model.TimePeriod;  
import com.example.TimeTracker.service.EmployeeService;  
import com.example.TimeTracker.service.TimePeriodService;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.websocket.server.PathParam;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@RestController  
@RequestMapping("/api")  
@CrossOrigin(origins = "http://localhost:3000")  
public class TimePeriodController {  
  
 private TimePeriodService timePeriodService;  
  
 public TimePeriodController(TimePeriodService timePeriodService) {  
 this.timePeriodService = timePeriodService;  
 }  
  
 @PostMapping("/addTime/{id}")  
 ResponseEntity<?> createTimePeriod(@RequestBody TimePeriod timePeriod, @PathVariable Long id) throws URISyntaxException {  
 if (timePeriod.getStartDate().getTime() > timePeriod.getEndDate().getTime()) {  
 return ResponseEntity.badRequest().build();  
 }  
 Employee employee = new Employee();  
 employee.setId(id);  
 timePeriod.setEmployee(employee);  
 timePeriodService.createTimePeriod(timePeriod);  
 return ResponseEntity.ok().build();  
 }  
  
 @GetMapping("/times/{id}")  
 List<TimePeriod> times(@PathVariable Long id) {  
 return this.timePeriodService.getAllTimePeriods()  
 .stream()  
 .filter(timePeriod -> timePeriod.getEmployee().getId() == id)  
 .collect(Collectors.toList());  
 }  
  
 @DeleteMapping("/time/delete/{id}")  
 ResponseEntity<?> deleteTimePeriod(@PathVariable Long id) {  
 timePeriodService.deleteTimePeriodById(id);  
 return ResponseEntity.ok().build();  
 }  
  
 @GetMapping("/times/year/{year}/{id}")  
 List<MonthWorkTimeDTO> times(@PathVariable Long id, @PathVariable int year) {  
 return this.timePeriodService.getYearWorkTimeReportByEmployeeId(id, year);  
 }  
  
  
}

package com.example.TimeTracker.dto;  
  
public class MonthWorkTimeDTO {  
  
 private String monthName;  
  
 private int percent;  
  
 private double hours;  
  
 public MonthWorkTimeDTO() {  
 }  
  
 public MonthWorkTimeDTO(String monthName, int percent, double hours) {  
 this.monthName = monthName;  
 this.percent = percent;  
 this.hours = hours;  
 }  
  
 public String getMonthName() {  
 return monthName;  
 }  
  
 public void setMonthName(String monthName) {  
 this.monthName = monthName;  
 }  
  
 public int getPercent() {  
 return percent;  
 }  
  
 public void setPercent(int percent) {  
 this.percent = percent;  
 }  
  
 public double getHours() {  
 return hours;  
 }  
  
 public void setHours(double hours) {  
 this.hours = hours;  
 }  
}

package com.example.TimeTracker.model;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
  
import javax.persistence.\*;  
import java.util.Date;  
import java.util.List;  
  
@Entity  
@Table(name="Employee")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.AUTO)  
 protected Long id;  
  
 private String firstName;  
  
 private String lastName;  
  
 private String email;  
  
 private Date dateOfBirth;  
  
 private String position;  
  
 @OneToMany(mappedBy="employee")  
 private List<TimePeriod> timePeriods;  
  
 public Employee() {};  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public List<TimePeriod> getTimePeriods() {  
 return timePeriods;  
 }  
  
 public void setTimePeriods(List<TimePeriod> timePeriods) {  
 this.timePeriods = timePeriods;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public Date getDateOfBirth() {  
 return dateOfBirth;  
 }  
  
 public void setDateOfBirth(Date dateOfBirth) {  
 this.dateOfBirth = dateOfBirth;  
 }  
  
 public String getPosition() {  
 return position;  
 }  
  
 public void setPosition(String position) {  
 this.position = position;  
 }  
}

package com.example.TimeTracker.model;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
  
import javax.persistence.\*;  
import java.util.Date;  
  
@Entity  
@Table(name="TimePeriod")  
public class TimePeriod {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.AUTO)  
 protected Long id;  
  
 private Date startDate;  
  
 private Date endDate;  
  
 @JsonIgnore  
 @ManyToOne  
 @JoinColumn(name="employee\_id", nullable=false)  
 private Employee employee;  
  
 public TimePeriod() {};  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public Date getStartDate() {  
 return startDate;  
 }  
  
 public void setStartDate(Date startDate) {  
 this.startDate = startDate;  
 }  
  
 public Date getEndDate() {  
 return endDate;  
 }  
  
 public void setEndDate(Date endDate) {  
 this.endDate = endDate;  
 }  
  
 public Employee getEmployee() {  
 return employee;  
 }  
  
 public void setEmployee(Employee employee) {  
 this.employee = employee;  
 }  
}

package com.example.TimeTracker.service.impl;  
  
import com.example.TimeTracker.model.Employee;  
import com.example.TimeTracker.repository.EmployeeRepository;  
import com.example.TimeTracker.service.EmployeeService;  
import org.springframework.stereotype.Component;  
  
import java.util.List;  
import java.util.Optional;  
  
@Component("EmployeeServiceImpl")  
public class EmployeeServiceImpl implements EmployeeService {  
  
 private EmployeeRepository employeeRepository;  
  
 public EmployeeServiceImpl(EmployeeRepository employeeRepository) {  
 this.employeeRepository = employeeRepository;  
 }  
  
 public Optional<Employee> getEmployeeById(Long id) {  
 return employeeRepository.findById(id);  
 }  
  
 public void deleteEmployeeById(Long id) {  
 employeeRepository.deleteById(id);  
 }  
  
 public Employee updateEmployee(Employee employee) {  
 return employeeRepository.save(employee);  
 }  
  
 public Employee createEmployee(Employee employee) {  
 return employeeRepository.save(employee);  
 }  
  
 public List<Employee> getAllEmployees() {  
 return employeeRepository.findAll();  
 }  
  
}

package com.example.TimeTracker.service.impl;  
  
import com.example.TimeTracker.dto.MonthWorkTimeDTO;  
import com.example.TimeTracker.model.Employee;  
import com.example.TimeTracker.model.TimePeriod;  
import com.example.TimeTracker.repository.EmployeeRepository;  
import com.example.TimeTracker.repository.TimePeriodRepository;  
import com.example.TimeTracker.service.TimePeriodService;  
import org.springframework.stereotype.Component;  
  
import java.util.\*;  
import java.util.stream.Collectors;  
  
@Component("TimePeriodServiceImpl")  
public class TimePeriodServiceImpl implements TimePeriodService {  
  
 private int MONTH\_WORK\_HOURS = 160;  
  
 private TimePeriodRepository timePeriodRepository;  
  
 public TimePeriodServiceImpl(TimePeriodRepository timePeriodRepository) {  
 this.timePeriodRepository = timePeriodRepository;  
 }  
  
 public Optional<TimePeriod> getTimePeriodById(Long id) {  
 return timePeriodRepository.findById(id);  
 }  
  
 public void deleteTimePeriodById(Long id) {  
 timePeriodRepository.deleteById(id);  
 }  
  
 public TimePeriod updateTimePeriod(TimePeriod timePeriod) {  
 return timePeriodRepository.save(timePeriod);  
 }  
  
 public TimePeriod createTimePeriod(TimePeriod timePeriod) {  
 return timePeriodRepository.save(timePeriod);  
 }  
  
 public List<TimePeriod> getAllTimePeriods() {  
 return timePeriodRepository.findAll();  
 }  
  
 public List<MonthWorkTimeDTO> getYearWorkTimeReportByEmployeeId(Long id, int year) {  
 List<TimePeriod> timePeriods = timePeriodRepository.findAll().stream().filter(timePeriod -> {  
 Calendar calendar = new GregorianCalendar();  
 calendar.setTime(timePeriod.getStartDate());  
 return ((calendar.get(Calendar.YEAR) == year) && (timePeriod.getEmployee().getId() == id));  
 }).collect(Collectors.toList());  
  
 Map<Integer, String> monthsMap = new HashMap<>();  
 monthsMap.put(0, "January");  
 monthsMap.put(1, "February");  
 monthsMap.put(2, "March");  
 monthsMap.put(3, "April");  
 monthsMap.put(4, "May");  
 monthsMap.put(5, "June");  
 monthsMap.put(6, "July");  
 monthsMap.put(7, "August");  
 monthsMap.put(8, "September");  
 monthsMap.put(9, "October");  
 monthsMap.put(10, "November");  
 monthsMap.put(11, "December");  
  
 List<MonthWorkTimeDTO> result = new ArrayList<>(11);  
  
 monthsMap.forEach((monthNumber, monthName) -> {  
 MonthWorkTimeDTO dto = new MonthWorkTimeDTO(monthName, 0, 0);  
 for (TimePeriod timePeriod : timePeriods) {  
 Calendar calendar = new GregorianCalendar();  
 calendar.setTime(timePeriod.getStartDate());  
 if (calendar.get(Calendar.MONTH) == monthNumber) {  
 double hours = (timePeriod.getEndDate().getTime() - timePeriod.getStartDate().getTime()) / (60 \* 60 \* 1000);  
 dto.setHours(dto.getHours() + hours);  
 }  
 }  
 //calculate percent of wor time in current month  
 dto.setPercent((int) ((dto.getHours() \* 100) / MONTH\_WORK\_HOURS));  
 result.add(dto);  
 });  
  
 return result;  
 }  
  
  
}

package com.example.TimeTracker.service;  
  
import com.example.TimeTracker.model.Employee;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface EmployeeService {  
  
 Optional<Employee> getEmployeeById(Long id);  
  
 void deleteEmployeeById(Long id);  
  
 Employee updateEmployee(Employee employee);  
  
 Employee createEmployee(Employee employee);  
  
 List<Employee> getAllEmployees();  
  
}

package com.example.TimeTracker.service;  
  
import com.example.TimeTracker.dto.MonthWorkTimeDTO;  
import com.example.TimeTracker.model.TimePeriod;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface TimePeriodService {  
  
 Optional<TimePeriod> getTimePeriodById(Long id);  
  
 void deleteTimePeriodById(Long id);  
  
 TimePeriod updateTimePeriod(TimePeriod timePeriod);  
  
 TimePeriod createTimePeriod(TimePeriod timePeriod);  
  
 List<TimePeriod> getAllTimePeriods();  
  
 List<MonthWorkTimeDTO> getYearWorkTimeReportByEmployeeId(Long id, int year);  
}

package com.example.TimeTracker;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class TimeTrackerApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(TimeTrackerApplication.class, args);  
 }  
  
}