Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РЕАЛИЗАЦИЯ НАСТОЛЬНОЙ ИГРЫ В ВИДЕ REAL-TIME WEB-ПРИЛОЖЕНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Голубевой Юлии Витальевны

Научный руководитель
доцент, к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

ВВЕДЕНИЕ

При проектировании современных web-приложений в настоящее время получили широкое распространение так называемые real-time технологии, которые позволяют автоматически обновлять изменившиеся данные. В подобных приложениях данные между клиентом и сервером передаются практически мгновенно, без существенных задержек. Для классических web-сайтов характерно то, что клиент не имеет представления о том, когда происходят изменения на серверной стороне, и вынужден запрашивать информацию вручную путем обновления страницы браузера. В приложениях реального времени, напротив, клиент всегда имеет самые актуальные данные.

В настоящее время не существует универсального способа создания real-time web-приложений. Некоторые методы просты в реализации, но уступают другим в эффективности. На выбор подхода влияют ожидаемая нагрузка на сервер, объем передаваемой информации, направление потока данных, а также максимальная допустимая задержка между появлением данных на сервере и передачей их клиенту.

Целью данной работы является реализация браузерной игры с применением real-time технологий для обновления содержимого web-страницы без участия пользователя. Для достижения этой цели были поставлены задачи:

- Изучение различных вариантов реализации real-time приложений;
- Изучение некоторых компонентов Spring Framework;
- Изучение средств для реализации технологии WebSocket в Java;
- Изучение библиотеки jQuery и плагина jQuery UI для добавления интерактивных элементов;
- Проектирование архитектуры приложения;
- Реализация web-сервера с применением real-time технологий;
- Верстка страниц сайта;
- Добавление визуальных эффектов с помощью библиотеку jQuery;
- Настройка взаимодействия между клиентом и сервером.

1 Средства, применяемые для разработки приложения

1.1 Real-time технологии

Для того, чтобы выбрать подходящую технологию для реализации обновления страниц в режиме реального времени, некоторые способы рассмотрены подробно.

1.1.1 Polling

Самое простое и понятное решение для непрерывного получения событий с сервера – это «частые опросы» (polling), то есть регулярные запросы на сервер с целью получения актуальных данных. В ответ на эти запросы сервер отправляет сообщения, которые содержат всю информацию, обновленную к моменту отправки. При этом, однако, возможна задержка между появлением и получением данных, в худшем случае на время, равное промежутку между двумя запросами.

1.1.2 Long Polling

Длинные опросы – усовершенствованная альтернатива частым опросам. Они также удобны в реализации, и при этом сообщения доставляются без задержек. Алгоритм взаимодействия следующий:

- Отправляется запрос на сервер;
- Соединение не закрывается сервером до появления сообщения;
- После появления сообщения сервер отвечает на запрос, отправляя данные;
- Браузер сразу же делает новый запрос.

1.1.3 Server Sent Events

Server Sent Events – стандарт, позволяющий браузерным клиентам получать поток обновлений с сервера по соединению HTTP, не обновляя страницу и не прибегая к опросу. Технология SSE представляет собой односторонний канал связи – поток событий от сервера к клиенту. Применение Server Sent Events оправдано, когда необходимо частое получение данных от сервера, но при этом запросы от клиента практически не поступают.

1.1.4 WebSocket

Протокол WebSocket представляет собой спецификацию, делающую возможным полнодуплексное подключение с использованием единственного сокета, по которому сообщения могут передаваться между клиентом и сервером. Применение протокола WebSocket позволяет отправлять сообщения как от клиента к серверу, так и наоборот, что избавляет от необходимости использовать опрос HTTP для двустороннего взаимодействия. Такой подход устраняет необходимость существования множества подключений для каждого клиента и снижает объем передаваемого трафика.

1.2 Серверные технологии

В то время, как для создания интерфейса web-страницы в простейшем случае достаточно одного html документа, серверная часть имеет гораздо более сложную конфигурацию. Необходимо обеспечить определенную структуру файлов приложения, подключить дополнительные модули и библиотеки, средства по работе с базами данных, настроить контейнер, в котором сервер будет запускаться и многое другое. Удобным и уже стандартным подходом для написания web-сервера на Java является использование Spring фреймворка.

1.2.1 Spring Framework

Spring представляет собой универсальный фреймворк с открытым исходным кодом для Java-платформы. Он применяется для обеспечения лучшей масштабируемости приложений, упрощает процесс тестирования и позволяет интегрироваться с другими библиотеками и фреймворками. Он содержит разнообразные модули, которые можно применять для решения большого количества различных задач. Основными являются:

- IoC (Inversion of Control) контейнер;
- AOP-фреймворк;
- Data Access фреймворк;
- MVC-фреймворк;
- Remote Access фреймворк;
- Testing-фреймворк.

MVC (Model View Controller) – это шаблон архитектуры приложений, основная идея которого заключается в разделении структуры проекта на несколько частей, где каждая отвечает за строго определенную функциональность. В

шаблоне MVC присутствуют три компонента:

— Модель. Представляет собой бизнес-данные, используемые в системе.
— Представление. Отвечает за отображение данных пользователю в выбранном формате, поддерживает взаимодействие с пользователями, отвечает за интернационализацию, оформление и т.д;
— Контроллер обрабатывает запросы на те действия, которые пользователь инициирует с помощью графического интерфейса, а также взаимодействует с уровнем обслуживания и обновляет представление.

Основная задача этого подхода – разделение пользовательского интерфейса и бизнес-логики. Строгое деление на модули делает код более структурированным, упрощает отладку и тестирование.

1.2.2 Spring WebSocket

Spring предлагает средства для реализации технологии WebSocket, а также STOMP в качестве подпротокола уровня приложения. За это отвечает компонент spring-websocket, который совместим со стандартом WebSocket API для Java (JSR-356).

1.2.3 STOMP

Работа напрямую с WebSocket (или SockJS) очень похожа на разработку веб-приложений с использованием только TCP сокетов. Без высокоуровневого сетевого протокола необходимо самостоятельно определять семантику сообщений, отправляемых между приложениями. Но работать с низкоуровневыми сокетами нет необходимости. Так же, как HTTP слой лежит поверх сокетов TCP, STOMP слой, основанный на фреймах, надстроен поверх WebSocket. На первый взгляд, STOMP сообщения выглядят очень похожими по структуре на HTTP запросы. Подобно HTTP запросам и ответам, фреймы STOMP состоят из команды, одного или нескольких заголовков и основной части.

В Spring предоставляется возможность работы со STOMP сообщениями с помощью программного модуля, основанного на Spring MVC. Конфигурация приложения начинается с аннотации `@EnableWebSocketMessageBroker`. Эта аннотация означает, что класс является не только WebSocket конфигурацией, но и брокером STOMP. Далее необходимо задать уникальный путь, по которому можно передавать и получать сообщения, переопределив метод `registerStompEndpoints()`. По этому пути изначально подключаются

все клиенты. Аналогом Spring MVC аннотации для обработки запросов от клиентов в Spring WebSocket приложениях является `@MessageMapping`. Метод, аннотированный `@MessageMapping`, обрабатывает только сообщения, поступающие на указанный адрес.

Для работы с WebSocket необходимо также настроить клиент на прием STOMP сообщений. Сначала нужно создать экземпляр SockJS соединения для заданного URL. Далее создается обертка над SockJS для отправки STOMP сообщений через WebSocket соединение. После этого используется STOMP клиент для установления соединения и передачи сообщения по указанному адресу.

1.2.4 Spring Boot

Не смотря на свою долгую историю существования, Spring Framework продолжает активно развиваться. Фреймворк охватывает новые области, такие, как облачные вычисления, big data, реактивное программирование и т.д. Одним из самых значительных нововведений стал модуль Spring Boot. Этот модуль предлагает новый подход к разработке Spring приложений, подразумевающий написание минимального количество вспомогательного кода. С помощью Spring Boot можно сконцентрироваться на реализации функциональности разрабатываемого приложения, затрачивая минимальное время на конфигурацию.

Spring Boot привносит много всего в разработку Spring приложений, но основными являются следующие функции :

- Automatic configuration. Предоставляет автоматическую конфигурацию, общую для многих Spring приложений;
- Starter dependencies. Средство, которое по используемой функциональности автоматически определяет все необходимые библиотеки и добавляет их;
- Command line interface. Эта функция Spring Boot помогает создавать и запускать проекты;
- Actuator. Помогает исследовать внутреннюю работу приложения.

1.2.5 JPA

Данные наряду с бизнес-логикой и пользовательскими интерфейсами представляют собой очень важную часть любого приложения. Основная до-

ля информации, которой пользуются приложения, хранится в базах данных. В реляционных базах данных информация хранятся в таблицах, связи между которыми реализуются путем использования внешних ключей и таблиц соединения.

В программах же, написанных на Java, происходит работа с объектами, которые являются экземплярами классов. Кроме классов также есть абстрактные классы, интерфейсы, перечисления, аннотации, методы, атрибуты, возможно применение сложных типов, наследование, полиморфизм.

Основная задача объектно-реляционного отображения (Object-Relational Mapping) в том, чтобы однозначно сопоставить таблицы баз данных и объекты языка. Это реализуется с помощью некоторой прослойки между этими двумя элементами – внешнего компонента или фреймворка.

Аннотации JPA позволяют настраивать таблицы, первичные ключи и столбцы. Аннотация `@Table` дает возможность изменять значения по умолчанию, связанные с определенной таблицей. Например, можно указать имя таблицы, в которой будут храниться данные, каталог и схему базы данных. Также можно задать ограничения с помощью аннотации `@UniqueConstraint` в сочетании с `@Table`. Аннотация `@Column` определяет свойства столбца. Можно изменить имя столбца (которое по умолчанию отображается в совпадающее с ним имя атрибута), а также указать размер и разрешить или запретить столбцу иметь значение null, быть уникальным. При создании сущности значение этого идентификатора может быть сгенерировано либо вручную с помощью приложения, либо автоматически с использованием аннотации @GeneratedValue.

Между двумя сущностями могут быть отношения «один к одному», «один ко многим», «многие к одному» или «многие ко многим». В JPA каждое соответствующее отображение именуется согласно типу связи источника и цели: аннотации @OneToOne, @OneToMany, @ManyToOne или @ManyToMany. Каждая аннотация может быть использована однонаправленным или двунаправленным путем.

1.2.6 JSP

Одним из способов реализации представления является технология JSP. JSP (JavaServer Pages) – это технология, которая упрощает разработку и сопровождение web-страниц, содержащих динамические данные. На странице JSP могут присутствовать элементы двух типов: статические исходные

данные, которые можно отобразить в одном из текстовых форматов HTML, SVG, WML, или XML, и JSP-элементы, конструирующие динамическое содержимое. Помимо того, возможно применение библиотеки JSP-тегов, а также EL (Expression Language), для внедрения Java-кода в статичное содержимое JSP-страниц. Прежде, чем JSP документ будет использован, компилятор JSP-страниц Jasper преобразует его в соответствующий сервлет. В свою очередь, сервлет пишется на языке Java и реализует определенный интерфейс. Важно понимать, что при этом сервлет также не является готовым приложением и не может функционировать самостоятельно, поэтому его необходимо поместить в соответствующий web-контейнер. Контейнер обеспечивает обмен данными между сервлетом и клиентами, берет на себя выполнение таких функций, как создание программной среды для функционирующего сервлета, идентификацию и авторизацию клиентов, организацию сессии для каждого из них.

1.3 Клиентская часть

Технологии HTML, CSS и JavaScript являются базовыми при создании web-страниц. Страницы сайта хранятся в виде HTML-кода в файловой системе сервера. После запроса к серверу клиент получает HTML-документ, этот файл при необходимости запрашивает дополнительные CSS и Javascript файлы, а затем браузер собирает все полученные элементы в одну страницу. Для упрощения разработки и повышения качества web-страниц создано большое количество фреймворков и библиотек.

1.3.1 jQuery UI

jQuery UI – это один из плагинов библиотеки jQuery. jQuery UI содержит достаточно большое количество эффектов, средств взаимодействия с пользователем и виджетов, которые делают создание web-приложений проще, а сами приложения более динамичными.

Плагин jQuery UI состоит из множества различных частей, которые можно разделить на три категории: виджеты, взаимодействия и эффекты.

1.3.2 Drag and drop

Передвижение элементов экрана является достаточно распространенным действием. Необходимости в этом нет, но такое действие может добавить странице удобство и динамичность. Плагин jQuery UI обеспечивает тип взаимо-

действия Drag and Drop с web-страницами с помощью виджетов Draggable и Droppable. Можно сделать перемещаемым как один элемент, так и применить класс к нескольким HTML-элементам, после чего использовать селектор класса и сделать сразу все элементы перемещаемыми.

В jQuery UI существует большое количество опций настройки для взаимодействия с виджетом Draggable. С помощью него можно контролировать направление, расстояние, на которое элемент можно передвигать, поведение при перемещении и другие дополнительные действия и параметры. Как и для любых других виджетов jQuery UI, элементу Draggable можно задать параметры, передав функции `draggable()` объект.

Отдельно виджет Draggable может быть полезен для диалоговых окон или других элементов страницы, которые можно перемещать по экрану, но сочетание его с виджетом Droppable позволяет создать интерактивные приложения, в которых перемещение одного элемента на другой может привести к некоторому результату. Виджет Droppable представляет собой «область бросания» для перемещаемых элементов. Когда элемент попадает в эту область, виджет Droppable может также активировать дополнительный программный код.

1.3.3 Resizable

Взаимодействие Resizable добавляет в элемент манипуляторы, путем перемещения которых пользователь может изменять размер компонентов. Некоторые браузеры автоматически предоставляют возможность изменения размеров для текстовых полей, но взаимодействие Resizable позволяет масштабировать абсолютно любые компоненты страницы. Описываемое взаимодействие также зависит от компонента Draggable. Помимо настроек, специфичных для самого компонента, также можно использовать настройки взаимодействия Draggable.

Одной из самых полезных опций при работе с взаимодействием Resizable является alsoResize. Благодаря этой опции можно определить те элементы, размеры которых также будут изменяться в одно время с размерами изменяемого элемента методом Resizable(). Данная опция удобна для обеспечения синхронного масштабирование элементов вместе с размерами их родителей.

2 Описание приложения

Разработанное приложение представляет собой реализацию настольной карточной игры «Бэнг». В игре участвуют 4 человека. Игроки ходят по очереди. В начале каждому игроку случайным образом назначается герой, обладающий такими характеристиками, как количество единиц здоровья, меткость и возможность уворота, а также имеющий уникальную способность, позволяющую ему совершать некоторые действия, недоступные остальным персонажам. Также каждый игрок получает роль, которая влияет на его стратегию и на необходимость взаимодействия с другими игроками. Цель игры – лишить здоровья других героев. Для этого разыгрываются карты действия, которые на каждом ходу пополняются из общей колоды, а также применяются способности персонажей.

Описание жизненного цикла приложения: При первом открытии сайта пользователь попадает на страницу авторизации. Страница содержит форму ввода логина и пароля, а также кнопки для входа и перехода к форме регистрации. На форме регистрации расположены поля для ввода логина, пароля и email адреса, а также кнопка, с помощью которой завершается регистрация. После того, как пользователь авторизуется, он попадет на главный экран, с которого можно начать игру, посмотреть правила, а также статистику предыдущих игр. Когда пользователь нажимает на кнопку поиска игры, он добавляется в очередь и ждет, пока в ней наберется достаточное количество человек, чтобы начать игру. После будет сконфигурирована новая игра, и пользователи попадут на игровую страницу.

Страница содержит игровое поле (колоды набора и сброса, а также область, где разыгрываются карты), панель с доступной информацией о других игроках (где отображены герои и их характеристики) панель самого игрока (где расположены карта героя, роль, характеристики и все имеющиеся карты действий), а также чат. Так как роли других игроков неизвестны, подразумевается, что игроки будут угадывать роли других персонажей по их поведению, а также, возможно, действовать сообща с другими участниками. Это невозможно без общения, поэтому на страницу и был добавлен игровой чат.

В свой ход, а также в ответ на действия других игроков, пользователь может перетащить карты на игровое поле, взять карты из колоды или сбросить лишние. После окончания игры пользователь снова попадает на главный экран.

2.1 Архитектура приложения

При проектировании архитектуры приложения был выбран шаблон Model View Controller. Это сделано по следующим причинам.

Структуру приложения можно условно разделить на три части. Первая часть – всё, что относится к игровой логике: классы, описывающие все сущности, участвующие в игре (пользователи, герои, их роли, способности, карты, колоды и т.д.), и методы, отвечающие за взаимодействие этих элементов, реализующие правила игры. Если вынести этот модуль отдельно, предоставив остальным частям приложения только интерфейсы для взаимодействия с ним, то при желании как-то изменить правила игры (добавив, например, новых персонажей или введя дополнительную роль) структура сервера, а также визуальные компоненты будут затронуты минимально.

Ко второй части относится браузерный клиент. MVC разделение позволяет разработать визуальную составляющую практически независимо от игровой логики, структуры сервера и протоколов взаимодействия.

Третья часть – контроллеры. Они являются связующим звеном между моделью и представлением, обеспечивают корректную передачу игровых данных браузеру, обрабатывают запросы, инициированные с помощью интерфейса пользователя, а также реализуют выбранный протокол взаимодействия с сервером. В данном приложении был выбран протокол STOMP, являющийся надстройкой над технологией webSocket. Но в случае, если вдруг понадобится сменить протокол, например, на обычные HTTP-запросы, то ни модель, ни представление не подвергнутся существенным изменениям.

2.2 Модель

Основным классом является класс Game. Он взаимодействует со всеми остальными структурными элементами. С помощью этого класса генерируется первоначальная конфигурация игрового стола, определяется очередность ходов, описывается порядок самих ходов, описываются условия окончания игры. Классы, описывающие карты, содержат информацию о карте, а также реализуют функциональность каждого типа карты в соответствии с правилами. Классы, описывающие герев, отвечают за создание героев, изменение характеристик, состояний, а также реализовывают способности в соответствии с правилами. Класс Pile описывает колоду, список карт, которые находятся в

сбросе, реализует методы перемешивание колоды, взятия из колоды и возврат карт в нее. Класс, Player описывает игрока, а именно: его героя, карты, роль, состояние пользователя, ассоциированного с ним, его положение в игре. Класс Table не реализует никакой функциональности, он представляет собой то, что должно быть видно пользователю, объединяя в себе элементы, описанные выше.

2.3 Контроллеры

Так как данные обновляются достаточно часто, и большое количество информации передается как от сервера клиенту (текущая конфигурация игрового стола), так и от клиента серверу (данные о ходе и сообщения чата), для обеспечения real-time обновлений экрана была выбрана технология webSocket. На серверной стороне для этих целей использовался Spring framework. Сервер имеет несколько контроллеров, настроенных на прием и отправку STOMP сообщений. Контроллер `@authorizationController` обрабатывает запросы по следующим путям:

— `/autorization.register`, отвечающий за регистрацию пользователей в системе;
— `/autorization.login`, отвечающий за вход в систему.

Ниже перечислены обрабатываемые запросы контроллером `@chatController`:

— `/chat.sendMessage` – отвечает за отправку сообщений;
— `/chat.addUser` – добавляет пользователя в чат;
— `/chat.removeUser` – удаляет игрока из чата.

Контроллер `@gameController` отвечает за следующие запросы:

— `/game.table` – возвращает текущую конфигурацию игры;
— `/game.start` – отвечает за начало игры;
— `/game.finish` – отвечает за окончание игры.

Контроллер `@mainController` обрабатывает главную страницу.

— `/main.rules` – отображает правила игры;
— `/main.stat` – отвечает за отображение статистики игр;
— `/main.search` – отвечает за поиск игры.

Основной метод, отвечающий за процесс игры, – `/game.table`. Как только кто-либо из пользователей совершил действие (например, выстрелил или восстановил единицу здоровья), клиент отправляет сообщение об этом событии на сервер. Контроллер принимает его, обращается к классам игровой

логики, и в ответ рассылает всем клиентам новую конфигурацию игрового стола с учетом изменений. Клиент принимает эту конфигурацию и отображает все изменения на экране.

Все методы условно разделены на два типа: одни настроены на отслеживание запросов пользователя (например, получение карты, которой сходил игрок), а другие – на передачу сообщений клиенту (к примеру, конфигурации стола). Метод `/game.search`, например, настроен только на прием сообщений. Он получает запрос от пользователя на добавление в очередь и реализует это, но не отправляет ничего в ответ. Код метода:

```
@MessageMapping("/main.search")
  public void search(@Payload ArrayList<User> queue,
   SimpMessageHeaderAccessor headerAccessor) {
      headerAccessor.getSessionAttributes()
      .put("username",
      queue);
      return;
  }
```

Данные о текущих играх хранятся в оперативной памяти сервера, так как информация о каждом сделанном ходе и сообщения чата не понадобятся уже после игры. Но информация о пользователях, а также результаты каждой игры для формирования статистики требуют постоянного хранения. Для этих целей использовалась база данных PostgreSQL, которая связана с объектами сервера посредством технологии JPA.

2.4 Представление

Все страницы созданы с помощью технологий HTML, CSS и JavaScript. С помощью плагина jQueryUI были реализованы некоторые интерактивные элементы. Для совершения действий с картами используется событие Drag and Drop. Чтобы сделать ход или сбросить карты, необходимо перетащить их в соответствующую область (центр стола или колода сброса). В конкретной игровой ситуации можно применять не все карты. Список доступных для хода карт приходит с сервера, и перемещаемыми являются только карты, находящиеся в этом списке, что не дает пользователю возможности сделать неверный ход.

Так как на экране расположено достаточно много элементов, они не слишком крупные, и, вероятно, пользователь может захотеть рассмотреть какую-либо из карт поближе. Для увеличения карт в размерах использовано событие Resizable. Необходимо потянуть карту за угол, чтобы её увеличить.

С помощью события Sortable можно расставить имеющиеся на руках карты в порядке, удобном пользователю. Перемещать можно все карты, но только в пределах площади, отведенной для их расположения, не выходя за границы контейнера. Как только перетаскиваемая карта наполовину пересечет соседнюю, они поменяются местами.

Клиент и сервер обмениваются между собой сообщениями в формате JSON. Для того, чтобы браузер мог получать STOMP сообщения по протоколу webSocket и обновлять содержимое web-страницы в режиме реального времени, необходимо сначала установить соединение.

Чтобы получить сообщение, не обновляя при этом страницу браузера, используется метод `subscribe()`, который отслеживает поступление данных по указанному адресу. В случае их появления вызывается метод, который обрабатывает полученные обновления и отображает их на экране. Каждый раз при необходимости изменения состояния экрана нужно модифицировать html-документ с помощью javaScript, удаляя и добавляя новые элементы, модифицируя и изменяя видимость существующих. Так, например, на странице игры происходит смена изображений на элементах, отвечающих за отображение карт, изменение характеристик героев, добавление и удаление карт, разыгрываемых в текущий ход, отображение сообщений чата. Метод реализует подписку на путь `/main.stat`. Как только завершается очередная игра, её результаты (уникальный идентификатор, участники и победитель) передаются по этому пути. Клиент при этом запускает метод, обрабатывающий JSON сообщение и изменяющий представление на экране. Для этого к элементу таблицы с помощью html тэгов добавляется строка, в которую встраивается только что полученная с сервера информация.

Таким образом, в реализованном приложении обеспечивается двунаправленное соединение клиента и сервера посредством технологии webSocket, а также производится динамическое изменение html-документа, что обеспечивает real-time обновление содержимого страниц приложения.

ЗАКЛЮЧЕНИЕ

В ходе данной работы было реализовано сетевое real-time приложение с браузерным клиентом. Для достижения этой цели были выполнены некоторые задачи, а именно:

- Изучение различных вариантов реализации real-time приложений;
- Изучение некоторых компонентов Spring Framework;
- Изучение средств для реализации технологии WebSocket в Java;
- Изучение библиотеки jQuery и плагина jQuery UI для добавления интерактивных элементов;
- Реализация web-сервера;
- Верстка страниц сайта;
- Добавление визуальных эффектов с помощью библиотеку jQuery;
- Настройка взаимодействия между клиентской и серверной частями.

После разработки приложения можно сделать выводы о том, что технология webSocket эффективна для real-time обновлений, MVC-шаблон хорошо подходит в качестве архитектуры web-приложения, фреймворк Spring удобен в применении и упрощает процесс разработки,а также, что библиотека jQuery предоставляет удобные методы для реализации визуальных эффектов.