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

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ ДЛЯ ОРГАНИЗАЦИИ И ПЛАНИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ЗАДАЧ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Удаловой Ольги Дмитриевны

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

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

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В рамках данной работы разработан web-планировщик, позволяющий отслеживать текущие дела пользователя, напоминать о необходимости выполнения, а также обладающий функцией автоматического распределения задач в соответствии с их приоритетами и сложностью.

Несмотря на то, что существует большое количество приложений для планирования и организации задач, многие оказывается слишком дорогими и громоздкими или не предоставляют возможность интеграции с уже существующими сервисами. Таким образом, разработка собственного органайзера, обладающего удобным функционалом, являющегося компактным и максимально гибким, остается актуальной.

В дальнейшем разработанное приложение будет использоваться как компактный, настраиваемый и бесплатный web-органайзер для управления задачами в IT-компании. Планируется интеграция с внутренними сервисами аутентификации, с почтой и системой выдачи рабочих задач. Для сотрудников, совмещающих работу с обучением в ВУЗе будет особенно актуальна функция автоматического распределения событий: она позволяет избежать ситуации наложения друг на друга нескольких задач по работе/учебе, когда приходится выбирать что-то одно, чтобы успеть в срок.

Таким образом, в рамках написания проекта для дипломной работы были поставлены следующие задачи:

1. Изучить структуру и механизмы взаимодействия между клиентом и сервером;
2. Рассмотреть наиболее популярные средства разработки и сделать выводы об их целесообразности;
3. Используя изученные технологии, реализовать web-приложение для планирования и организации пользовательских задач.

Структура и объем работы. Бакалаврская работа состоит из введения, трех разделов, заключения, списка использованных источников и одного приложения. Общий объем работы — 51 страница, из них 44 страницы — основное содержание, включая 18 рисунков, цифровой носитель в качестве приложения, список использованных источников информации — 20 наименований.

1 Краткое содержание работы

1.1 Первый раздел работы

Раздел посвящен рассмотрению теоретических основ Web-технологий: архитектуре «Клиент-сервер». Это тип сетевой архитектуры, в рамках которого вычислительные мощности и сетевая нагрузка распределены между серверами — некими программными компонентами, выполняющими сервисные функции — и клиентами, запрашивающими «услуги» сервера. И клиент, и сервер являются программным обеспечением, причем чаще всего они находятся на различных рабочих станциях и взаимодействуют друг с другом удаленно посредством сетевых протоколов. Т.к. к серверу, как правило, обращается множество клиентов, он размещается на отдельной особым образом настроенной станции с высоким уровнем производительности.

Приложение, реализующее клиент-серверную архитектуру, называют web-приложением.

1.2 Второй раздел работы

Раздел посвящен рассмотрению типовой структуры Web-приложения, а также обзору используемых при разработке технологий.

На рисунке 1 приведена общая схема работы Web-приложения.

Серверная часть приложения: уровень хранения данных. В этом подразделе описано схематичное устройство уровня хранения данных в web-приложении, а также используемые технологии — СУБД PostgreSQL и фреймворк Hibernate.

Уровень хранения данных включает в себя DataSource (или источник данных) и Persistence layer, где организован непосредственно доступ к данным.

Сразу стоит упомянуть такое понятие, как DAO — Data Access Object. Это один из широко используемых шаблонов проектирования — объект, предоставляющий доступ к источнику данных и покрывающий его дополнительным слоем абстракции.

В Persistence layer определяется один или несколько интерфейсов, благодаря которым вышестоящим Business Layer будет иметь доступ к данным. Несмотря на обилие всевозможных DataSource, наиболее «классическим» решением является использование реляционных баз данных.

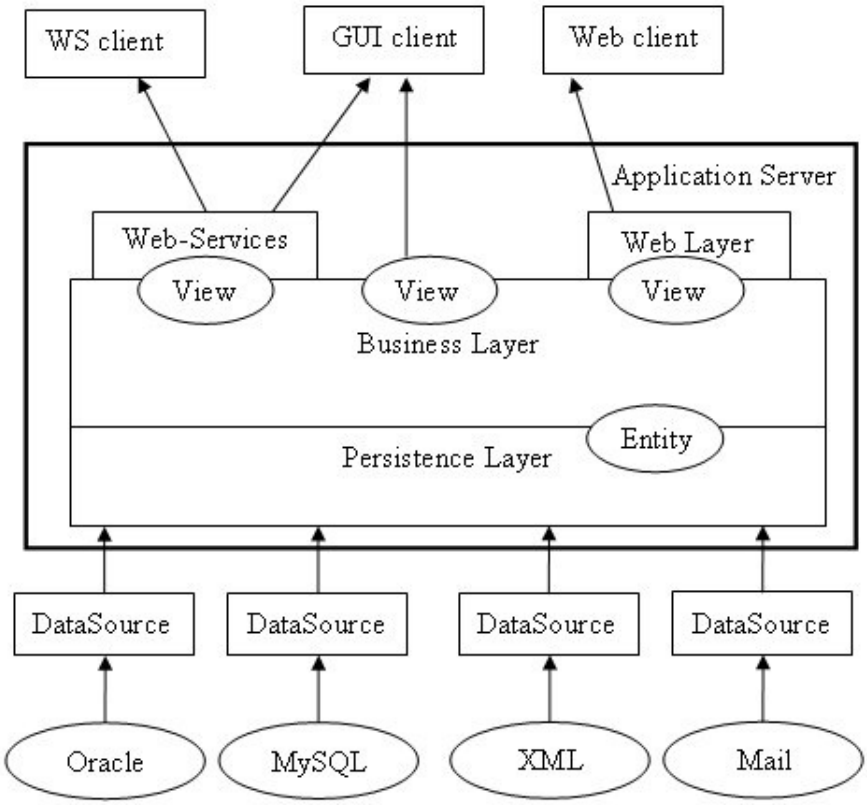

Рисунок 1 – Внутренняя архитектура Web-приложения

В рамках рассматриваемой ВКР в качестве системы управления базой данных была выбрана PostgreSQL — бесплатная СУБД с открытым исходным кодом

Также, для упрощения работы с базой данных использовался фреймворк Hibernate. Hibernate реализует технологию ORM: она позволяют работать с записями в таблице как с обычными Java-объектами: для этого классы-сущности (классы, экземпляры которых представляют собой записи таблицы) помечаются аннотациями Hibernate.

Серверная часть приложения: уровень бизнес-логики. В этом подразделе описано схематичное устройство уровня бизнес-логики в web-приложении, а также используемые технологии — язык программирования Java и фреймворк Spring.

Основной задачей этого уровня является, по сути, обработка информации, полученной из Persistence Layer — добавляется определенная логика, в соответствии с которой данные изменяются и обновляются. Главной задачей разработчика на этом уровне является организация работы с данными в виде

цельных транзакций, а также разработка оптимальной иерархии сущностей.

Инициализация иерархии классов, как правило, производится с использованием такого паттерна, как Inversion of Control («Инверсия контроля») или, сокращенно, IoC. Основная его идея заключается в возложении задачи определения связей между классами и объектами на некий фреймворк. При разработке серверной части приложения и уровя бизнес-логики в частности использовался язык программирования Java и универсальный фреймворк Spring.

В качестве средства веб-разработки используется Java Enterprise Edition. Она предоставляет набор интерфейсов API для разработки и запуска портируемых, надежных, масштабируемых и безопасных серверных приложений [1].

Для упрощения разработки и реализации шаблона «Инверсия контроля» использовался фреймворк Spring Boot. Это фреймворк, разработанный на основе Spring Framework. Он, по сути, представляет из себя «обертку» над Spring, состоящую из дополнительных конфигураций и технологий.

Web-уровень: взаимодействие клиента и сервера. В этом подразделе описано взаимодействие приложения и Web-клиента. В рамках ВКР при разработке использовалась технология веб-сокетов(WebSockets).

WebSockets — это относительно молодая технология по взаимодействию клиента(браузера) и сервера в режиме реального времени. Взаимодействие асинхронно: в отличие от классических HTTP-запросов веб-сокеты умеют работать с двунаправленным потоком информации. Клиент может отправить одно сообщение и ожидать отклика. То есть клиент как бы «слушает» сервер, которые отправляет сообщения по мере готовности.

Клиентская часть приложения. В этом подразделе описано устройство клиентской части приложения, а также рассмотрены такие технологии как язык программирования JavaScript, язык разметки HTML и таблицы стилей CSS.

JavaScript — это мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Основная задача языка - это добавление интерактивности на страницы вебсайта. Скрипты подключаются к HTML-страницам и выполняются при загрузке — JS не требуется компилировать.

В контексте разработки клиентской части веб-приложения основной задачей JavaScript является манипулирование элементами DOM-модели веб-станицы [2]. В качестве инструмента для создания веб-страниц и их последующего отображения в браузере используется HTML — язык разметки гипертекста.

CSS — это язык описания внешнего вида документа разметки. В основном используется для оформления веб-страниц, для задания цветов, шрифтов, расположения элементов на странице и т.п.

Основной целью CSS является отделение описания внешнего вида документа от описания его логической структуры [3]: это упрощает разработку, повышает гибкость и дает возможность управлять структурой и представлением по отдельности. Также, это минимизирует объем кода и позволяет избегать повторов.

1.3 Третий раздел работы

Раздел посвящен разработке собственного web-приложения на практике. Описываемое приложение представляет из себя web-органайзер, отслеживающий задачи, напоминающий пользователю о необходимости их выполнения, а также обладающей функцией автоматического распределения задач с плавающей датой в зависимости от уже заданных событий.

Структура разработанного приложения. Подраздел посвящен описанию архитектуры разработанного приложения. Она разработана в соответствии с описанной ранее типовой структурой:

- Уровень хранения и управления данными, где происходит вся работа с информацией из источника.
- Уровень бизнес-логики — здесь реализована основная логика серверной части приложения, а также обработка данных, полученных на предыдущем уровне.
- Веб-уровень — взаимодействие серверной и клиентской частей приложения.
- Интерфейс пользователя — клиентская часть приложения.

На рисунке 2 приведена диаграмма классов разработанного приложения.

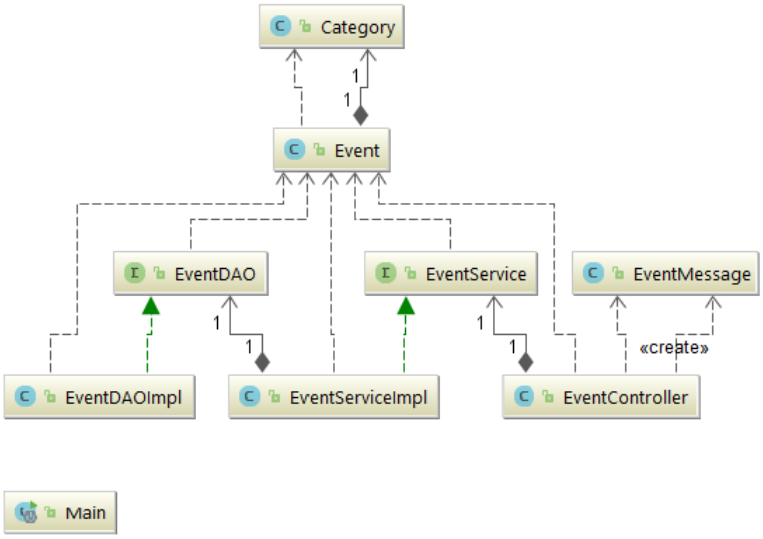

Рисунок 2 – Диаграмма классов разработанного приложения

Работа с базой данных. Подраздел посвящен описанию механизму обработки исходных данных из базы PostgreSQL. В качестве средства по работе с базой использовался фреймворк Hibernate. Разработана структура базы данных, состоящая из двух таблиц: таблица events, содержащая всю информацию о пользовательских событиях, а также таблица categories, в которой хранятся все возможные категории вышеописанных событий. Таблицы связаны отношением один-ко-многим: пользователь может определять множество событий одной категории, но у одного события может быть только одна категория.

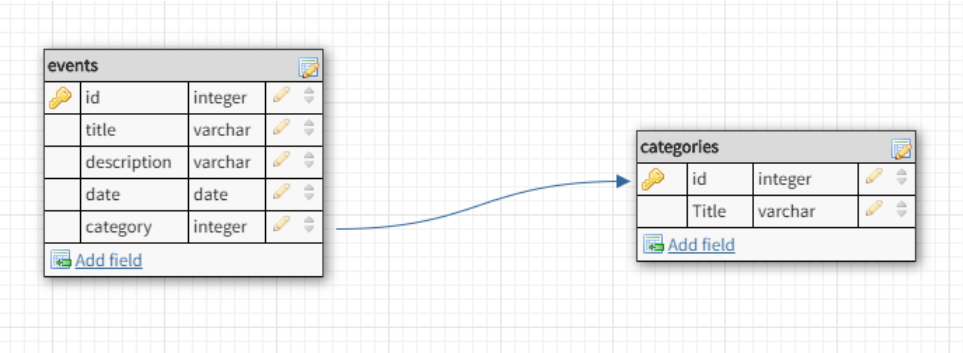

Рисунок 3 – Схема используемой в приложении базы данных

Средства Hibernate позволяют работать с записями в таблице как с обычными Java-объектами: для этого классы-сущности (классы, экземпляры которых представляют собой записи таблицы) помечаются аннотациями Hibernate. Далее классы-сущности оборачиваются слоем обработки данных, или DAO-слоем. Здесь находятся объекты, предоставляющие доступ к источнику данных

и покрывающие его дополнительным слоем абстракции. Вся работа непосредственно с базой данных — создание сессий транзакций, выполнение запросов, запись новых объектов реализуется только через DAO-объект.

Разработка бизнес-логики. В данном подразделе описывается бизнес-логика приложения, взаимодействие классов и основные алгоритмы.

Поверх DAO-слоя из уровня хранения данных располагается сервисный слой. Именно на уровне бизнес-логики реализован механизм автоматической расстановки событий с плавающей датой в зависимости от уже прописанных в базе. Если пользователь включил эту функцию, то в методе `public void eventAutoCreation` класса `EventServiceImpl` система выбирает свободные дни и заполняет базу «предположениями» о том или ином событии. Пользователь может подтвердить предположение или удалить его.

Взаимодействие клиента и сервера. В данном подразделе описан механизм работы веб-уровня.

Здесь находится класс-контроллер, который принимает запросы, приходящие от сервера, перенаправляет их на обработку в сервисные классы, и отправляет ответы серверу.

После получения сообщения о, например, новом событии, управление отдается классу EventService, который, в свою очередь, взаимодействуя с EventDAO создает объект в базе. Далее контроллер должен создать сообщение для отправки на клиент, в котором будет вся информация о созданном событии — сообщение, тело которого — JSON-объект, в котором содержится вся информация о передаваемом событии.

Для задания структуры сообщения создан простой Java-класс с необходимыми текстовыми полями в качестве параметров: EventMessage. Механизм обмена сообщениями реализован средствами библиотеки spring-messaging.

Непосредственная «точка входа» в приложение — класс Main. Здесь запускается приложение спринг бут и происходит настройка обмена сообщений через WebSockets

На клиентской части для подключения WebSockets разработан скрипт WebSockets.js: здесь открывается соединение, и после успешного подключения клиент «подписывается» на /topic/events: на этот адрес сервер будет отсылать

сообщения с пользовательскими событиями.

Разработка клиентской части приложения. В данном подразделе описан интерфейс пользователя разработанного приложения.

ри разработке клиентской части приложения использовался язык программирования JavaScript в сочетании с языком разметки HTML и таблицами стилей CSS.

Основная часть интерфейса — календарь с отображенными на нем значками задач. Он условно разбит на три части: заголовок, в котором выводятся текущие месяц и год, непосредственно календарь с днями недели и обозначением сегодняшней даты и легенда по цветам — каждой категории событий соответствует свой цвет.

Сначала создаются статические элементы — блок напоминаний в левой части экрана и заготовка для календаря — блочный элемент с определенным айди. После этого запускается скрипт, создающий объект Calendar, который в качестве полей содержит список событий, html-элемент, где календарь будет выводиться(блочный элемент calendar, заранее заданный в index.html) и указатель на сегодняшний день. Для работы с датами использовалась библиотека moment.js.

Развертывание приложения в контейнере сервлетов. В данном подразделе описаны технологии, при помощи которых происходит развертывание приложения в контейнере. Для этого использовался фреймворк Maven и контейнер сервлетов Tomcat.

Maven — фреймворк для автоматической сборки. Он создает конечный файл проекта на основе его спецификации в конфигурационных файлах. Полученный в результате сборки файл развертывается на встроенном в Spring Boot контейнере сервлетов Tomcat. Таким образом, в клиент-серверной модели в роли клиента выступает по-прежнему браузер, а сервер заменяет программное обеспечение на той же машине — Apache Tomcat.

ЗАКЛЮЧЕНИЕ

В ходе работы были выполнены поставленные цели: подробно рассмотрена архитектура типичного web-приложения, сделан обзор используемых фреймворков и библиотек, проведен их анализ. Реализована серверная часть приложения с использованием вышеупомянутых технологий, организована работа с базой данных, а также клиент-серверное взаимодействие с использованием технологии WebSockets.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Эккель, Б.* Философия Java / Б. Эккель. — Санкт-Петербург: Питер, 2017.

2 *Флэнаган, Д.* JavaScript: Подробное руководство. 6-е издание / Д. Флэнаган. — Москва: Символ, 2016.

3 *Хеник, Б.* HTML и CSS. Путь к совершенству. / Б. Хеник. — Санкт-Петербург: Питер, 2011.