# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 4](#_Toc104211330)

[ВВЕДЕНИЕ 5](#_Toc104211331)

[1 ПОСТАНОВКА ЗАДАЧИ 7](#_Toc104211332)

[1.1 Актуальность поставленной задачи 7](#_Toc104211333)

[1.2 Анализ требований задачи 7](#_Toc104211334)

[1.3 Комментарии к логике приложения 8](#_Toc104211335)

[1.3.1 Атрибуты заявки 8](#_Toc104211336)

[1.3.2 Жизненный цикл заявки 9](#_Toc104211337)

[2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 12](#_Toc104211338)

[2.1 Описание системы 12](#_Toc104211339)

[2.2 Проектирование базы данных 13](#_Toc104211340)

[3 РАЗРАБОТКА ПРИЛОЖЕНИЯ 18](#_Toc104211341)

[3.1 Реализация серверной части 18](#_Toc104211342)

[3.1.1 Технологии 18](#_Toc104211343)

[3.1.2 Комментарии к логике приложения при реализации 19](#_Toc104211344)

[3.1.3 Комментарии к реализации 21](#_Toc104211345)

[3.2 Тестирование 23](#_Toc104211346)

[ЗАКЛЮЧЕНИЕ 26](#_Toc104211347)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc104211348)

# ВВЕДЕНИЕ

В настоящее время в компании используется разработанное ранее приложение Vacation Planner, предназначенное для планирования отпусков сотрудников. В связи с тем, что сотрудники совмещают работу с учебой и, как и в выборе дней отпуска, заранее планируют дату, стоимость, а также утверждают с руководителем и администрацией возможность посещать курсы и тратить время на учебу в указанный промежуток времени, возникла потребность разработать схожее с Vacation Planner приложение для планирования обучения сотрудников. Компании было бы удобнее, если процесс планирования обучения сотрудников был бы автоматизирован.

Для разработки приложения планирования обучения сотрудников необходимо разработать новый модуль с новой функциональностью.

Актуальность данной работы заключается в том, компании необходимо приложение, которое поможет структурировать и облегчить процесс работы с заявками на обучение, поэтому данное приложение поможет автоматизировать процесс планирования и реализации организуемого компанией обучения ее сотрудников.

Цель данной работы заключается в разработке серверной части модуля планирования обучения сотрудников. Необходимо разработать функциональность, реализующую планирование обучения и отображение общего плана обучения (совокупности утвержденных заявок).

Приложение должно реализовывать возможность создания и изменения заявки; рассмотрение заявки сотрудниками, имеющими права руководителя или администратора, или участника группы, отвечающей за рассмотрение заявок на обучение, после рассмотрения заявка может быть утверждена или отклонена заявки, а также возможность мониторинга всех заявок в указанном диапазоне дат и по интересующим критериям. Данные или статистику можно экспортировать в формате EXCEL.

Требуется настроить систему оповещений сотрудников о начале создания заявок, завершения создания заявок, начале курсов, а также оповещение конкретных сотрудников при условии, что требуется уточнение заявки.

# ПОСТАНОВКА ЗАДАЧИ

В рамках проекта создания модуля планирования обучения сотрудников разрабатывается отдельное Java EE приложение.

Серверная часть приложения должна реализовывать заданный функционал. Задача ограничивается разработкой серверной логики и REST API. REST API для обмена данными должен использовать JSON.

## Актуальность поставленной задачи

В настоящее время в компании используется разработанное ранее приложение Vocation Planner, реализующее организацию процесса планирования отпусков сотрудников, но нет приложения, позволяющего планировать обучение сотрудников. На основе существующего приложения возникла идея о создании нового модуля, который позволил бы автоматизировать процесс планирования и организации обучения сотрудников компании.

## Анализ требований задачи

Рассмотрим необходимые требования к реализации данной системы.

Разработанный модуль должен реализовывать логику:

* создание, изменение и рассмотрение заявок на обучение,
* выдача данных в заданном диапазоне дат – курсы для отдельного сотрудника, для группы, или для всех сотрудников. Выдача всех, запланированных или пройденных курсов.
* выдача статистики:

1. распределение числа обучающихся сотрудников по датам в заданном диапазоне дат;
2. распределение числа курсов обучения по сотрудникам в заданном диапазоне дат;

* экспорт данных или статистики за выбранный год в формат EXCEL. Должна быть возможность фильтрации экспортируемых данных,
* отправка оповещений сотрудникам и ответственным за рассмотрение заявок при приближении сроков начала создания заявок, их рассмотрения, начала обучения.

## Комментарии к логике приложения

Все сотрудники компании принадлежат определенной группе. В зависимости от того, какой группе принадлежит сотрудник, определяется его роль в компании, а также круг возможностей работы с заявками.

У каждой группы есть руководитель. Исключается случай, когда группы образуют иерархию, то есть руководитель имеет руководителя. Также сотрудник не может иметь более 1 руководителя.

Для работы с заявками на обучение вводится группа «Corp. Education Team», члены которой ведут процесс реализации заявок (в том числе – рассматривают их от имени Администрации). Входящие в нее сотрудники, включая Директора, имеют одинаковые права по работе с заявками.

### Атрибуты заявки

При создании заявки сотрудник должен указать такие поля, как:

* название курса (обязательное поле),
* ссылка на курс (обязательное поле),
* компания – поставщик курса (обязательное поле),
* область обучения (SW Dev, QA, SW Architect, PM, PO, Processes, и так далее, может расширяться при необходимости) (обязательное поле),
* тип: обучение или конференция (обязательное поле),
* обоснование заявки (обязательное поле),
* формат: очный или дистанционный (обязательное поле),
* начало обучения (обязательное поле),
* длительность обучения (обязательное поле),
* стоимость обучения (обязательное поле),
* ссылка на отзыв (необязательное поле),
* комментарий (необязательное поле).

### Жизненный цикл заявки

Заявка на обучение в рамках жизненного цикла проходит несколько этапов. Заявка имеет атрибут «статус», который показывает ее текущее состояние и может иметь следующие значения (их множество может расширяться при необходимости):

* создано,
* изменено,
* требуется уточнение,
* утверждено руководителем,
* отклонено руководителем,
* утверждено администрацией,
* отклонено администрацией,
* перенесено,
* отменено,
* оплачено,
* начато,
* завершено.

Жизненный цикл заявки:

1. создание заявок,
2. изменение заявок,
3. рассмотрение заявок,
4. ограничение на доступ к заявкам (область видимости),
5. оповещение.

Описание каждого этапа представлено в таблице (таблица 1).

Таблица 1 – Описание этапов жизненного цикла

|  |  |
| --- | --- |
| Название этапа жизненного цикла | Описание этапа жизненного цикла |
| 1 | 2 |

Продолжение таблицы 1

|  |  |
| --- | --- |
| 1 | 2 |
| Создание заявок | Любой сотрудник (включая руководителя) может создать одну или несколько заявок на обучение в заданном году (задается в файле ресурсов приложения).  В заявке сотрудник должен указать параметры, указанные в атрибутах заявки (см. п. 1.3.1).  Все доступные ему параметры обязательные, и должны иметь значения.  Создание заявок на обучение доступно только в заданный промежуток времени (задается в файле ресурсов). |
| Изменение заявок | Если заявка еще не была рассмотрена руководителем (утверждена или отклонена), сотрудник может изменить заявку (все доступные ему параметры) и вновь отправить на утверждение.  Если заявка уже была утверждена, сотрудник может перенести заявку (изменить ее даты). |
| Рассмотрение заявок | Рассмотрение заявок доступно только в заданный промежуток времени (задается в файле ресурсов).  До утверждения или отклонения, заявка может быть возвращена сотруднику для уточнения, после этого он может отредактировать ее и вновь отправить на утверждение.  Заявка последовательно утверждается Руководителем и Администрацией. После этого она может быть только перенесена сотрудником. |
| Ограничение на доступ к заявкам | Сотрудник имеет доступ ко всем созданным им заявкам (и только к ним).  Руководитель имеет доступ ко всем заявкам, созданным сотрудниками его группы, включая его самого.  Члены группы «Corp Education Team» имеют доступ ко всем заявкам сотрудников. |

Продолжение таблицы 1

|  |  |
| --- | --- |
| 1 | 2 |
| Оповещение | Оповещение отсылается письмом на адрес, указанный для сотрудника в БД.  Оповещение содержит текст и атрибуты курса - название, ссылка на курс, дата начала и ссылка для получения данных курса через REST API. |

Требуемые оповещения описаны в таблице 2

Таблица 2 – Описание логики оповещений

|  |  |
| --- | --- |
| Когда | Получатель |
| 1 | 2 |
| За 2 недели до начала периода создания заявок. | Все сотрудники. |
| За день до начала периода создания заявок. | Все сотрудники. |
| За день до окончания периода создания заявок. | Все сотрудники. |
| В день начала периода рассмотрения заявок. | Руководители. |
| За день до окончания периода рассмотрения заявок. | Руководители. |
| После запроса на уточнение заявки. | Сотрудник, если требуется уточнение заявки. |
| Ежедневно. | Руководители, имеющие не рассмотренные заявки после окончания срока рассмотрения. |
| За 1 месяц до начала курса. | Администрация. |
| За 1 неделю до начала курса. | Сотрудник. |
| За 1 день до начала курса. | Сотрудник. |

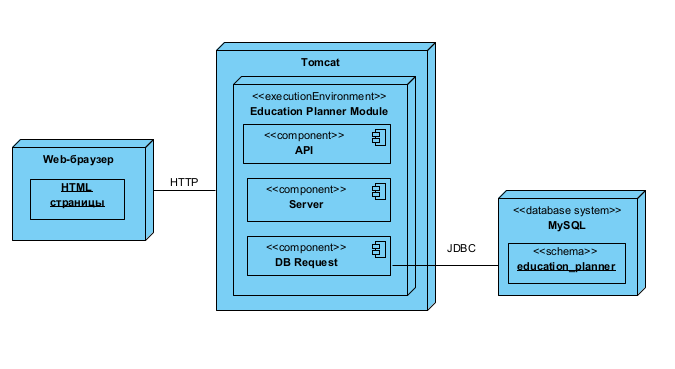
# ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## Описание системы

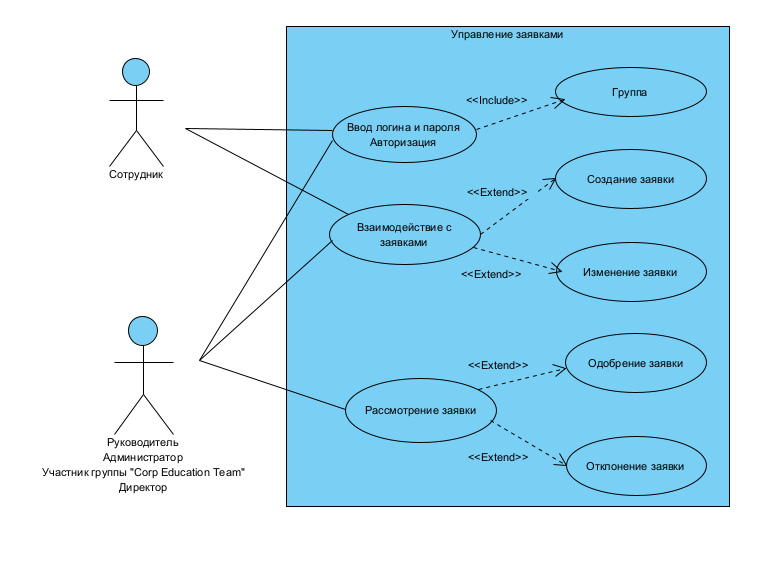
Моя задача заключается в реализации серверной части приложения.

На рисунке 1 изображена системная архитектура приложения. На изображении представлен:

* Web-браузер (клиент), выступающий в роли представления доступного функционала,
* сервер приложений. Слой, исполняющий основные правила бизнес-логики и обеспечивающий связь между клиентом и базой данных,
* сервер базы данных. Обеспечивает хранение и извлечение информации.

Рисунок 1 – Системная архитектура приложения

В связи с тем, что у каждого пользователя есть своя роль, а значит и возможности над заявками ограничены, то для отображения действий авторизованного пользователя в зависимости от его роли использую UML-диаграмму Use Case (рисунок 2).

Рисунок 2 - UML-диаграмму Use Case

После авторизации сотрудника проверяется, какой группе он принадлежит. В зависимости от его роли определяется то, что он может делать с заявками.

Обычный сотрудник после авторизации имеет возможность создать и изменить заявку. Руководитель, администратор, директор могут не только создавать свои заявки, но и одобрять или отклонять заявки сотрудников.

## Проектирование базы данных

Одним из этапов проектирования архитектуры является проектирование базы данных. База данных предназначена для хранения информации. На рисунке 3 подставлена диаграмма данных.

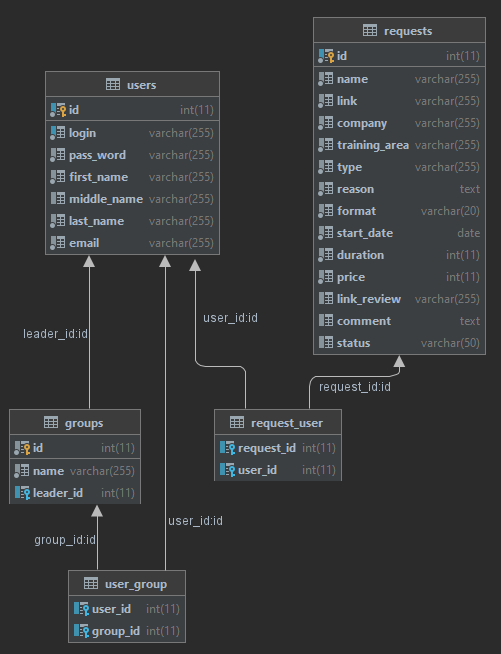


Рисунок 3 – Диаграмма данных

Основные сущности таблицы:

* Users – сотрудники,
* Groups – группы,
* Users\_group – сотрудник-группа,
* Requests – заявки,
* Request\_user – заявки-пользователь.

Сущность Users (пользователи или, в данном случае, сотрудники) хранит в себе информацию о сотруднике. С помощью этой сущности осуществляется авторизация пользователя, проверка на корректность данных при входе в систему. Также храниться адрес электронной почты, используемый для рассылки уведомлений сотрудникам. Структура сущности Users представлена в таблице 3.

Таблица 3 – Структура сущности Users (сотрудники)

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип данных | Описание |
| 1 | 2 | 3 |
| id (PK) | INT | Идентификационный номер сотрудника. |
| login | VARCHAR (255) | Логин сотрудника для авторизации в системе. |
| pass\_word | VARCHAR (255) | Пароль сотрудника для авторизации в системе. |
| first\_name | VARCHAR (255) | Имя сотрудника. |
| middle\_name | VARCHAR (255) | Отчество сотрудника (если оно у него есть). |
| last\_name | VARCHAR (255) | Фамилия сотрудника. |
| email | VARCHAR (255) | Электронная почта сотрудника. |

Сущность Groups (группы) хранит в себе информацию о группах. Используя данную сущность, можно узнать роль сотрудника в кампании, а также руководителя группы. Структура сущности Groups представлена в таблице 4.

Таблица 4 – Структура сущности Groups

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип данных | Описание |
| 1 | 2 | 3 |
| id (PK) | INT | Идентификационный номер группы. |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| name | VARCHAR (255) | Название группы, по которой определяется роль сотрудника и возможный круг действий с заявками. |
| leader\_id (FK) | INT | Идентификационный номер руководителя группы, информацию о котором можно найти, в сущности, users по полю id. |

Сущность Users\_group (пользователи-группа) содержит в себе информацию о том, какой группе принадлежит каждый сотрудник, отношение сущностей Users и Users\_group осуществляется по полям id (сущность Users) и user\_id (сущность Users\_group).

Структура сущности Users\_group представлена ниже в таблице 5.

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип данных | Описание |
| 1 | 2 | 3 |
| user\_id (PK) | INT | Идентификационный номер сотрудника. Связь с сущностью users. |
| group\_id (FK) | INT | Идентификационный номер группы. Связь с сущностью groups. |

Сущность Requests (заявки) содержит поля, отвечающие за атрибуты заявки. Хранит в себе всю информацию, описание заявки, которую заполнил пользователь.

Структура сущности Requests представлена в таблице 6

Таблица 6 – сущность Requests

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип данных | Описание |
| 1 | 2 | 3 |
| id (PK) | INT | Идентификационный номер заявки. |
| name | VARCHAR (255) | Название курса, который хочет посетить сотрудник. |
| link | VARCHAR (255) | Ссылка на курс, который хочет посетить сотрудник |
| company | VARCHAR (255) | Название компании, принадлежащей курс, который планирует посетить сотрудник. |

Продолжение таблицы 6

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| training\_area | VARCHAR (255) | Область обучения. |
| type | VARCHAR (255) | Тип: обучение или конференция. |
| reason | TEXT | Обоснование заявки. |
| format | VARCHAR (20) | Формат обучения: очный или дистанционный (online/offline). |
| start\_date | DATE | Начало обучения. |
| duration | INT | Продолжительность обучения. |
| price | INT | Цена обучения. |
| link\_review | VARCHAR (255) | Ссылка на курс, опциональное поле. Значение по умолчанию NULL. |
| comment | VARCHAR (255) | Комментарий от сотрудника. Опциональное поле, значение по умолчанию NULL. |
| status | VARCHAR (50) | Статус заявки. |

Сущность Request\_user (заявка-пользователь). Содержит в себе информацию о статусе заявки, сотруднике, который создал заявку. По статусу можно отслеживать состояние заявки. Отношение между сущностями Request и Request\_user осуществляется по полям request\_id (Request\_user) и id (Requests), а между сущностями User и Request\_user по полям id и user\_id соответственно.

Структура сущности Requests представлена в таблице 7

Таблица 7 – сущность Request\_user

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип данных | Описание |
| 1 | 2 | 3 |
| request\_id (PK) | INT | Идентификационный номер заявки. |
| user\_id (FK) | INT | Идентификационный номер сотрудника. |

# РАЗРАБОТКА ПРИЛОЖЕНИЯ

Перед началом разработки приложения было проделано несколько подготовительных шагов, чтобы лучше разобраться в архитектуре и продумать план действий.

В первую очередь было необходимо установить необходимое программное обеспечение и сделать необходимые действия перед началом проекта:

* установить приложение Vocation Planner, приложение для планирования отпусков сотрудников, которое используется в компании. Необходимо было установить для того, чтобы ознакомиться с существующей логикой приложения,
* необходимо было спроектировать базу данных для планирования обучения. В компании уже есть база данных, используемая для планирования отпусков сотрудников, но было необходимо добавить еще 2 сущности для работы с заявками на обучение,
* разработать каркас приложения и слоев (слой API | Слой бизнес-логики | Слой доступа к БД),
* в слое бизнес-логики надо было продумать логику создания и изменения заявок на обучение,
* продумать логику изменения состояния заявок,
* продумать логику выдачи и фильтрации данных,
* продумать логику выдачи статистических данных,
* продумать логику экспорта данных в формате EXCEL,
* продумать логику создания и отправки оповещений,
* продумать, как тестировать каждый этап.

## Реализация серверной части

### Технологии

Для разработки приложения по заданию используется язык Java.

Обычно Java EE приложения делят на три уровня:

* клиентский,
* средний уровень,
* уровень доступа к данным.

1. Клиентский уровень представляет из себя некоторое приложение, которое запрашивает данные у Java EE сервера (среднего уровня). Сервер, в свою очередь, обрабатывает запрос клиента и возвращает ему ответ. Клиентским приложением может быть браузер, отдельное приложение (мобильное либо десктопное) или другие серверные приложения без графического интерфейса.
2. Серверный уровень подразделяется, в свою очередь, на web-уровень и уровень бизнес-логики.

На web-уровне используются такие технологии Java EE:

* Java Server Pages (JSP),
* Servlets.

Уровень бизнес-логики состоит из компонент, в которых реализована вся бизнес-логика приложения. Бизнес-логика — это код, который обеспечивает функциональность, покрывающую нужды некоторой конкретной бизнес-сферы. Данный уровень можно считать ядром всей системы.

1. Уровень доступа к данным. К этому уровню за данными обращается уровень бизнес-логики.

На данном уровне используется технология:

* Java Database Connectivity API (JDBC).

Для разработки и развертывания приложения использую Tomcat 9.

Для работы с Базой данных используется MySQL 5.7.

### Комментарии к логике приложения при реализации

В первую очередь нужно было реализовать логику авторизации пользователя в приложении для определения его роли в компании. В зависимости от роли пользователя, он будет обладать набором возможностей по взаимодействию с заявками. На рисунках х и х изображены примеры аутентификации пользователя.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок х – Ввод пользовательских данных

Изображение выглядит как текст

Автоматически созданное описание

Рисунок x – Страница в роли администратора

В случае неверно введённых данных страница авторизации обновляется.

На рисунке х приведена схема полной логики приложения, начиная с авторизации пользователя.

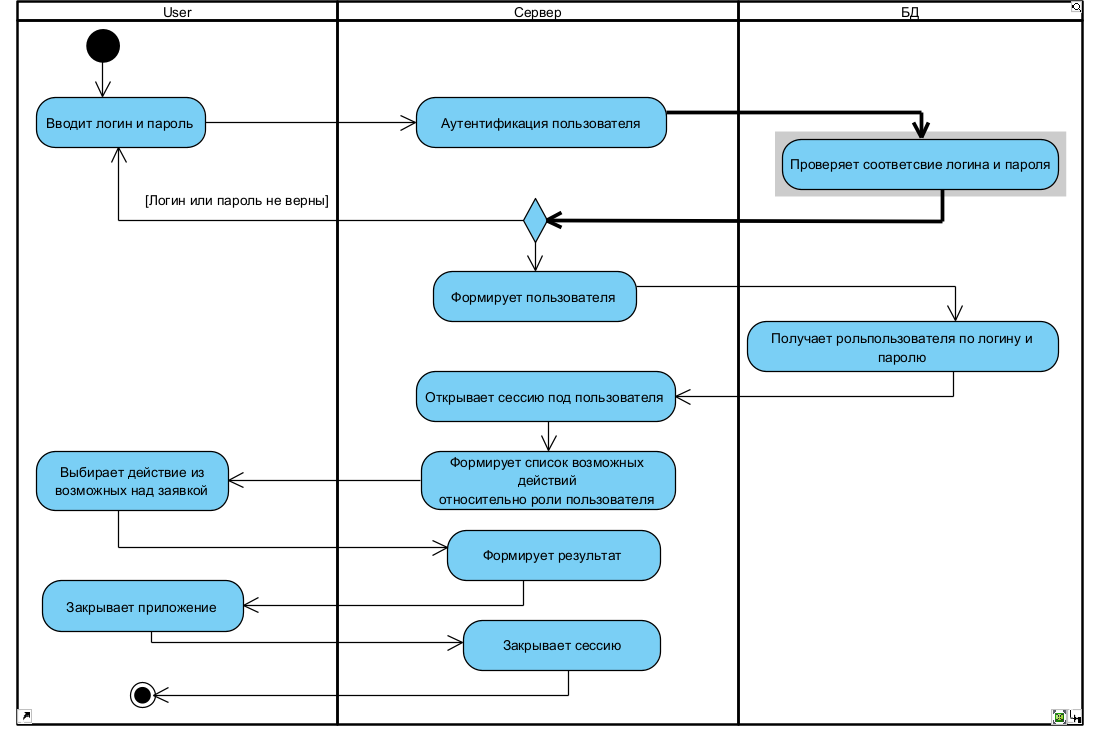


Рисунок х – логика приложения

### Комментарии к реализации

На web-уровне серверной части приложения используются Servlets. REST API должен использовать JSON для обмена данными, в связи с этим я использую Jackson - библиотека Java JSON, позволяющую сериализовать объекты Java в JSON и десериализовать объекты JSON в объекты Java (рисунки x, x).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок x – Объект, передаваемый в формате JSON

Изображение выглядит как текст, снимок экрана, внутренний, черный

Автоматически созданное описание

Рисунок x – Сериализация объекта в строку

Также для уменьшения количества кода и улучшения читаемости использую библиотеку Lombok (рисунок x).

На уровне доступа к базе данных использую библиотеку JDBC. Здесь с помощью запросов к базе данных получаю результаты и информацию, которую в дальнейшем использую в слое с бизнес-логикой (рисунок х).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок х – Слой соединения с базой данных

## Тестирование

Тестирование производилось поэтапно, в процессе разработки.

С помощью web-браузера или программы Postman, проверяла визуализацию приложения, get и post запросы (рисунки х и х).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок х – Проверка Get и Post запросов, формирование результатов в JSON

Изображение выглядит как текст

Автоматически созданное описание

Рисунок x – Проверка визуализации для post запросов через web-браузер

Для тестирования слоя соединения с базой данной, я создала тестовую базу данных, аналогичную основной. Корректность запросов проверяла на тестовой базе данных, как показано на рисунке х.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок х – Тестирование взаимодействия с базой данных

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы реализована серверная часть модуля планирования обучения сотрудников, предназначенная для автоматизации процесса планирования обучения.

Данная работа была выполнена в 5 этапов:

* + анализ существующего в компании приложения для планирования отпусков сотрудников,
  + анализ требований для модуля планирования обучения сотрудников,
  + проектирование системы,
  + разработка серверной части приложения,
  + тестирование.

Идея данного приложения возникла в связи с тем, что сотрудники компании часто проходят курсы для повышения квалификации или изучения нового, а для этого также требуется утверждение сроков, стоимости, занятости у руководителей и администрации компании. Из-за того, что в компании уже существует приложение, позволяющее автоматизировать процесс отслеживания заявок на отпуска от сотрудников, возникла идея также автоматизировать процесс планирования и реализации организуемого компанией обучения ее сотрудников.

В мою задачу входило написание только серверной логики приложения, включающей в себя как проектирование базы данных для хранения информации о пользователях, заявках, статусах заявки, так и разработка бизнес-логики приложения.

Полученное приложение позволяет создавать заявку, изменять. В зависимости от роли сотрудника в компании – одобрять или отклонять заявку. Сотрудники также могут экспортировать данные за выбранный год в формате EXCEL, при необходимости данные могут отфильтровать.

Настроена система оповещений сотрудников о начале создания заявок, завершения создания заявок, начале курсов, а также оповещение конкретных сотрудников при условии, что требуется уточнение заявки.

Поставленные задачи были выполнены, цель выпускной квалификационной работы достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Что такое Apache? Подробный обзор веб-сервера Apache [Электронный ресурс] – URL: <https://www.hostinger.com.ua/rukovodstva/web-server-apache/#Apache_%D0%B8%D0%BB%D0%B8_Tomcat>
2. Платформа Java 2 Enterprise Edition [Электронный ресурс] – URL: <https://intuit.ru/studies/courses/64/64/lecture/1890?page=2>
3. Упражнение 3.2. Создание вариантов использования [Электронный ресурс] – URL: <http://sp.cs.msu.ru/ooap/exerb2018.html>
4. Требования к оформлению ВКР [Электронный ресурс] – URL: <https://student.itmo.ru/files/1314>
5. JPA работа с базой данных [Электронный ресурс] – URL: <https://devcolibri.com/jpa-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D0%B1%D0%B0%D0%B7%D0%BE%D0%B9-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-%D1%87%D0%B0%D1%81%D1%82%D1%8C-1/>
6. Метод executeQuery. Получение данных [Электронный ресурс] – URL: <https://metanit.com/java/database/2.5.php>
7. Топ сайтов с серверной частью на Java EE [Электронный ресурс] – URL: <https://sohabr.net/habr/post/220757/?version=35021>
8. Язык программирования Java [Электронный ресурс] – URL: <https://web-creator.ru/articles/java>
9. Введение в Java EE – Javarush [Электронный ресурс] – URL: <https://javarush.ru/groups/posts/2637-vvedenie-v-java-ee>
10. Создаём веб-приложение с Java Servlets [Электронный ресурс] – URL: <https://tproger.ru/translations/building-a-web-app-with-java-servlets/>
11. Apache Tomcat 9 (Version 9.0.62, Mar 31, 2022) [Электронный ресурс] – URL: <https://tomcat.apache.org/tomcat-9.0-doc/windows-service-howto.html>