Geekbrains

Веб-разработка приложения на Java с использованием фреймворка Spring-Boot для учета данных работников на предприятии.

Программа: Разработчик

Специализация Веб-разработка на Java.

Кузнецов Василий Николаевич

Красноярск

2025 год

Содержание

[Содержание](https://docs.google.com/document/d/1t649czIfxsCa6KUteMlApNLSZY5xbZe9/edit" \l "heading=h.7mt9x4b37ce)

[Введение](https://docs.google.com/document/d/1t649czIfxsCa6KUteMlApNLSZY5xbZe9/edit" \l "heading=h.1fob9te)

[Теоретическая и практическая главы](https://docs.google.com/document/d/1t649czIfxsCa6KUteMlApNLSZY5xbZe9/edit" \l "heading=h.hst8x8f1r8tz)

[Заключение](https://docs.google.com/document/d/1t649czIfxsCa6KUteMlApNLSZY5xbZe9/edit" \l "heading=h.dhwmmf3wq1kk)

[Список используемой литературы](https://docs.google.com/document/d/1t649czIfxsCa6KUteMlApNLSZY5xbZe9/edit" \l "heading=h.m8jvbwhv7ujg)

[Приложения](https://docs.google.com/document/d/1t649czIfxsCa6KUteMlApNLSZY5xbZe9/edit" \l "heading=h.s6k7t2lwcp0k)

Благодарю компанию Geekbrains в полном составе за предоставленную возможность окунуться в свои 48 лет в мир разработки!!!!

Введение

**Глава 1. Разработчик и приложения**

1.1 Что такое веб-приложение

1.2 Что такое разработка веб-приложений, кто такой разработчик.

1.3 Этапы разработки веб-приложений

1.4 Источники требований: что такое техническое задание и как проводить разработку, если техническое задание отсутствует

**Глава 2. . Разработка веб-приложения на Java**

2.1 Что такое Java

2.2 **Некоторые особенности языка Java**

**2.3 Фреймворки**

**2.4 Фреймворк Spring**

**2.5 Основные модули Spring**

**2.6 Maven**

**2.7 Docker**

Глава 3. **Практическая часть**

Глава 4. Заключение

Глава 5. Список используемой литературы

Приложения

Глава 1. Разработчик и приложения

* 1. Что такое веб-приложение

**Веб-приложение** — это **программное обеспечение, которое**

**работает на веб- сервере**.

Доступ к нему можно получить по сети, в браузере. 

Веб-приложение позволяет пользователям взаимодействовать с данными и функционалом приложения, хранящимися на сервере. Оно создаётся для того, чтобы решать задачи клиента и задачи бизнеса: например, для управления проектами или онлайн-покупок.

**Основные характеристики веб-приложения**:

* **Интерактивность** — возможность взаимодействия пользователя с приложением.
* **Доступность** — веб-приложение может быть запущено на любом устройстве с доступом к интернету.
* **Масштабируемость** — способность приложения эффективно работать при большом числе пользователей.
* **Обновляемость** — возможность внесения изменений и обновления приложения без необходимости переустановки на устройствах пользователей.
* **Безопасность** — защита приложения от несанкционированного доступа и атак.

Веб-приложения широко используются в различных сферах, включая медицину, банковское дело, электронную коммерцию и многие другие

* 1. Что такое разработка веб-приложений, кто такой разработчик

**Разработка веб-приложений** — это процесс создания интерактивных и многоуровневых сайтов, объединённых с системой управления клиентскими данными, видеоплатформами, онлайн-калькулятором и другими сервисами.

**Веб-разработчик** — это специалист, который создаёт новые и

поддерживает уже существующие сайты. Он помогает заказчику

решить задачу оптимальным способом, пишет код, тестирует его и

оценивает результаты. 

**Некоторые задачи веб-разработчика**:

**Фронтенд-разработчик** занимается клиентской частью сайта —

той, которую пользователь видит в браузере. Например, может

разработать функцию корзины в интернет-магазине.

**Бэкенд-разработчик** создаёт серверную часть сайта и работает с базами данных и функциями для обработки информации на сервере. Может отвечать за авторизацию, обработку платежей, работу с почтовыми уведомлениями. — «полный набор»)

**Фулстек-разработчик** (от англ. full stack — «полный набор») занимается и клиентской, и серверной частью сайта. Участвует во всём цикле веб-разработки и знаком с основными языками, технологиями и фреймворками как для фронтенда, так и для бэкенда.

1.3 **Основные этапы разработки веб-приложений**:

**Аналитика**. Определение требований и планирование, сбор информации о функциональности, дизайне, требованиях к безопасности.

**Дизайн**. Проектирование UX, разработка интерфейса до получения макетов всех страниц и элементов.

**Бэкенд**. Серверная логика программы, разработка основы её функционирования.

**Фронтенд**. Интерфейсы, иногда — часть логики программы, разработанная и реализованная на фронте.

**Тестирование**. Исследование работоспособности, отслеживание, исправление ошибок в коде и дизайне.

**Запуск**. Развёртывание проекта на боевом сервере, публикация в web.

**Поддержка**. Отслеживание и поддержание работоспособности.

**1.4 Источники требований: что такое техническое задание и как проводить разработку, если техническое задание отсутствует**

**Техническое задание (ТЗ)** — это документ, в котором фиксируются требования к проекту. Он определяет цель, структуру, свойства и методы какого-либо проекта, а также помогает команде и заказчику понять, каким продукт должен получиться в итоге.

**Если техническое задание отсутствует**, для проведения разработки можно использовать следующие подходы:

**Техническое проектирование**. Эта стадия предшествует написанию кода. В ходе неё закрываются все нерешённые вопросы по требованиям, дизайну и API, а также прорабатываются сложные технические моменты.

**Функциональные схемы**. Это схемы, представляющие собой блочную структуру функций приложения с навигационными переходами. Их готовит команда дизайна в тандеме с аналитиком.

**Подход «Contract First»**. В нём описывают интерфейс (контракт) между различными компонентами системы ещё до начала реализации этих компонентов. Контракт определяет, как компоненты будут взаимодействовать друг с другом, какие данные будут передавать и какие операции будут доступны.

**Глава 2. Разработка веб-приложения на Java**

2.1 Что такое Java

**Java** — **строго типизированный объектно-ориентированный язык программирования общего назначения**. Был создан Джеймсом Гослингом в 1995 году в компании Sun Microsystems.

2.2 **Некоторые особенности языка Java:**

**Объектно-ориентированность**. Каждый компонент программы (объект) содержит данные и методы для их обработки. Это помогает создавать модульные и легко поддерживаемые приложения, повторно использовать код в разном ПО.

**Кроссплатформенность**. Способность работать на разных операционных системах без изменений в исходном коде. Это сделала возможным Java Virtual Machine (JVM), которая интерпретирует и выполняет байт-код Java для запуска на любой платформе, поддерживающей JVM.

**Многопоточность**. Язык поддерживает одновременное выполнение нескольких задач внутри одного приложения. Это особенно полезно для многозадачных и высоконагруженных программ: веб-серверов и баз данных.

**Сильная типизация**. Каждая переменная должна иметь определённый тип данных, и любые операции над ними проверяются на соответствие типов. Это позволяет предотвращать множество ошибок на этапе компиляции, что делает код стабильнее.

**Управление памятью через механизм сборки мусора**. Язык автоматически управляет выделением и освобождением памяти, что снимает с программистов бремя ручного управления.

**Java используется** для создания мобильных и веб-приложений, корпоративного программного обеспечения, устройств Интернета вещей (IoT), игр, больших данных, распределённых и облачных приложений и др..

**2.3 Фреймворки**

**Фреймворк** (с англ. framework — «каркас, структура») — **это набор ИТ-инструментов, который упрощает разработку и поддержку сложных и высоконагруженных веб-проектов**.

**Фреймворки делятся по типам задач**:

**Фронтенд-фреймворки**. Отвечают только за видимую часть сайта или приложения: отрисовку интерфейса, постраничную навигацию. Популярные фронтенд-фреймворки: Angular, Next.js, Vue.js.

**Бэкенд-фреймворки**. Отвечают за внутреннюю (серверную) часть сайтов и приложений: хранение, обработку и анализ данных. Популярные бэкенд-фреймворки: Django, Flask — для Python, Laravel — для PHP, Spring — для Java, Ruby on Rails — для Ruby, Asp.NET — для C#.

**Фулстек-фреймворки**. Отвечают одновременно за внешнюю и внутреннюю часть работы сайтов и приложений. Популярные фулстек-фреймворки: Meteor.js — для JavaScript, Kivy — для Python, Xamarin — для C#.

Выбор фреймворка влияет на скорость, удобство разработки, производительность, масштабируемость и поддержку приложения.

**2.4 Фреймворк Spring**

**Spring** — это **фреймворк с открытым исходным кодом для языка программирования Java**. Он был создан для упрощения разработки и поддержки масштабируемых, слабосвязанных и повторно используемых приложений.

Spring предоставляет набор инструментов и библиотек, которые упрощают и ускоряют процесс разработки, позволяя сосредоточиться на бизнес-логике приложения.

**С помощью Spring Java-разработчики создают серверное ПО** для работы с современными десктопными, мобильными и веб-приложениями. Spring полезен как для простых сервисов, так и для больших проектов из множества модулей и сотен функций

**Spring — это набор взаимосвязанных мини-фреймворков**, созданных для работы над разными частями приложения. Подключать их можно по отдельности, в зависимости от задач.

**2.5 Основные модули Spring**

**Spring Core**. Ядро фреймворка, которое включает контейнер зависимостей и обеспечивает инверсию управления (IoC) и аспектно-ориентированное программирование (AOP).

**Spring MVC**. Модуль для построения веб-приложений с использованием модели MVC. Он включает широкий спектр веб-компонентов, таких как контроллеры, перехватчики, валидаторы и др..

**Spring Data**. Модуль, который предоставляет шаблоны для работы с различными системами хранения данных, такими как реляционные базы данных, NoSQL базы данных, кеширование и поисковые системы.

**Spring Security**. Модуль для защиты приложения на уровне аутентификации и авторизации, обеспечивает возможность реализации системы контроля доступа к ресурсам приложения.

**Spring Boot**. Модуль, который упрощает процесс настройки и развёртывания Spring-приложений, предоставляя автоматическую конфигурацию и встроенные зависимости для быстрого старта проекта.

**Spring Cloud**. Модуль, который помогает разработчикам работать с различными облачными сервисами и платформами. Он предоставляет инструменты для настройки, мониторинга, маршрутизации, автоматического масштабирования, безопасности и других характеристик облачно-ориентированных приложений.

**Spring Integration**. Модуль, который упрощает интеграцию Spring-приложений с другими системами и сервисами через абстракции на уровне кода и конфигурации.

**Spring Batch**. Модуль, который обеспечивает работу с пакетной обработкой данных, позволяя эффективно обрабатывать большие объёмы данных и обмениваться ими между различными системами.

**Spring Test**. Модуль, который помогает разработчикам писать и выполнять юнит-тесты для Spring-приложений, предоставляя инструменты для тестирования компонентов, конфигурации и интеграции с другими системами.

Аннотации в Spring — это специальные метки, которые используются для управления поведением компонентов в приложении на основе этого фреймворка. Они упрощают конфигурацию и настройку приложения, избавляя разработчиков от необходимости вручную писать большое количество конфигурационного кода.

В своем приложении я использовал следующие аннотации:

@Configuration используется для обозначения класса, который содержит методы, определяющие бины (объекты), управляемые Spring. Каждой метод может быть помечен аннотацией @Bean, чтобы указать, что он создает и конфигурирует экземпляр бина.

@Bean в Spring используется для определения метода, который будет создавать и управлять объектом (или "бином") в контексте Spring. Каждый раз, когда Spring нуждается в экземпляре этого бина, он вызовет указанный метод и предоставит возвращаемый объект

@RestController в Spring используется для определения компонента контроллера, который обрабатывает HTTP-запросы и отвечает на них, возвращая данные в формате, удобном для работы с RESTful API. Эта аннотация является комбинацией двух аннотаций: @Controller и @ResponseBody.

@Autowired чаще всего используется в классах, помеченных такими аннотациями, как @Controller, @Service, @Repository, и @Component.

@GetMapping в Spring используется для обработки HTTP GET-запросов. Это специализированная версия аннотации @RequestMapping, которая упрощает создание RESTful веб-сервисов, предоставляя явный способ указать, что метод контроллера должен обрабатывать только запросы типа GET.

@PostMapping в Spring используется для обработки HTTP POST-запросов. Это специализированная версия аннотации @RequestMapping, которая упрощает создание RESTful веб-сервисов, предоставляя явный способ указать, что метод контроллера должен обрабатывать только запросы типа POST.

@RequestBody в Spring используется для связывания содержимого HTTP-запроса с объектом в Java. Это особенно полезно, когда данные передаются в теле запроса, например, в формате JSON или XML. @RequestBody позволяет автоматически десериализовать данные из запроса и преобразовать их в объект, который затем можно использовать в методе контроллера.

@DeleteMapping в Spring используется для обработки HTTP DELETE-запросов. Это специализированная версия аннотации @RequestMapping, предоставляющая удобный способ управления удалением ресурсов на сервере. Эта аннотация делает код более понятным и лаконичным, указывая, что метод контроллера предназначен для удаления.

@PathVariable **в Spring используется для извлечения значений из URL-параметров в методах контроллеров. Она позволяет связывать часть URL с параметрами метода, что делает возможным динамическое получение данных, указанных в запросе.**

@PutMapping **в Spring используется для обработки HTTP PUT-запросов. Как и другие аннотации, такие как** @GetMapping**,** @PostMapping**, и** @DeleteMapping**,** @PutMapping **предоставляет более удобный и читаемый способ работы с определенными HTTP-методами в RESTful веб-серверах.**

@RequestParam **в Spring используется для извлечения параметров запроса из URL в методах контроллеров. Она позволяет получать значения, переданные через параметры запроса, и использовать их в качестве аргументов методов, что делает API более гибким и удобным для взаимодействия.**

@Entity **в Java используется в контексте объектно-реляционного отображения (ORM) и является частью библиотеки Java Persistence API (JPA). Она используется для обозначения класса как сущности, которая представляет собой таблицу в реляционной базе данных. Сущности являются основными компонентами, которые позволяют разработчикам работать с базой данных, используя объектный подход.**

@Table **в Java используется в контексте объектно-реляционного отображения (ORM) и является частью Java Persistence API (JPA). Она применяется для указания, как класс-сущность отображается на таблицу в реляционной базе данных.**

**@Id — это аннотация в Java, используемая в контексте Java Persistence API (JPA) для обозначения первичного ключа сущности в базе данных. Она указывает, что поле, к которому применяется аннотация, является уникальным идентификатором записи в таблице.**

**@SequenceGenerator — это аннотация в Java Persistence API (JPA), используемая для определения генератора последовательностей, который создает уникальные значения для первичных ключей в базе данных. Она позволяет задать параметры, такие как имя генератора, имя последовательности, начальное значение и шаг увеличения**

**@GeneratedValue — это аннотация в Java Persistence API (JPA), которая указывает, что значение поля, помеченного как первичный ключ с аннотацией @Id, будет автоматически генерироваться. Она может использоваться с различными стратегиями генерации, такими как IDENTITY, SEQUENCE или TABLE, в зависимости от используемой базы данных и требований к созданию ключей.**

**\**

@Service **в Java используется в рамках Spring Framework и является частью концепции Spring для создания слоев сервиса в приложении. Она служит для обозначения компонента сервисного слоя, который содержит бизнес-логику приложения.**

**2.6 Maven**

**Apache Maven** — **инструмент для автоматизации сборки проектов** на Java и других языках программирования. Он помогает разработчикам правильно подключить библиотеки и фреймворки, управлять их версиями, выстроить структуру проекта и составить к нему документацию.

Одна из главных особенностей фреймворка — **декларативное описание проекта**. Это значит, что разработчику не нужно уделять внимание каждому аспекту сборки — все необходимые параметры настроены по умолчанию. Изменения нужно вносить лишь в том объёме, в котором программист хочет отклониться от стандартных настроек.

Ещё одно достоинство проекта — **гибкое управление зависимостями**. Maven умеет подгружать в свой локальный репозиторий сторонние библиотеки, выбирать необходимую версию пакета, обрабатывать транзитивные зависимости.

При необходимости систему сборки можно настроить под собственные нужды, используя готовые плагины и архетипы.

**2.7 Docker**

**Docker** — это **программная платформа для разработки, доставки и запуска контейнерных приложений**. Она позволяет создавать контейнеры, автоматизировать их запуск и развёртывание, управляет жизненным циклом. С помощью Docker можно запускать множество контейнеров на одной хост-машине.

Платформа Docker представлена в двух вариантах: бесплатная Community Edition под лицензией Apache и платная Enterprise Edition. Платная версия предназначена для коммерческого использования и распространяется по проприетарной лицензии.

**Docker контейнер** — это стандартизированный, изолированный и портативный пакет программного обеспечения, который включает в себя всё необходимое для запуска приложения, включая код, среду выполнения, системные инструменты, библиотеки и настройки.

Благодаря контейнеризации и использованию Docker разработчики могут сосредоточиться на создании и использовании приложений, не думая о базовой операционной системе или системе развёртывания.

Глава 3. **Практическая часть**

Разработку нашего приложения будем проводить в программе **IntelliJ IDEA Community Edition 2024.** Свое приложение я назвал Employee(сотрудник).

Как я думаю оно пригодится в любой компании для учета сотрудников.

Разработку будем проводить с помощью фреймворка Spring –Boot.

Для создания проекта зайдем на сайт [https://start.spring.io](https://start.spring.io/).

Выбираем следующее:

Проект Maven, Язык Java, Версия Spring-Boot 3.4.1 на момент начала написания проекта она была актуальна, сейчас уже 3.4.2. заполняем **Project Metadata.** Packaging файл архив с проектом jar, версия java 17. А также зависимости:

* **Spring Web Web**Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**Spring Web** — это зависимость, которая предоставляет **контейнер сервлетов Apache Tomcat** (является дефолтным веб-сервером). Проще говоря, сервлеты — это классы, которые обрабатывают все входящие запросы

Также Spring Web содержит модуль **Web**, который обеспечивает такие функции, как загрузка файлов.

* **Spring Data JPA SQL**Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

**Spring Data JPA** **упрощает работу с JPA** **путём автоматизации написания повторяющегося кода**. Этот фреймворк построен на основе JPA и предлагает упрощённое API для исполнения CRUD-операций и создания запросов на основе имён методов.

Некоторые особенности Spring Data JPA:

**Реализация шаблона «репозиторий»**. Это позволяет создать чёткое разделение слоя данных, делая код более сжатым и управляемым.

**Усовершенствованная система именования методов для запросов**. Запросы создаются на основе наименований методов, что экономит время и обеспечивает типобезопасный доступ к данным без дублирования SQL или JPQL-кода.

**Динамическая обработка данных: пагинация и сортировка**. Spring Data JPA эффективно обрабатывает большие объёмы данных, делая приложения более отзывчивыми и производительными.

**Аудит и проверка запросов для обеспечения надёжности**. Spring Data JPA предоставляет возможности для аудита и валидации запросов, что значительно повышает качество и надёжность приложения.

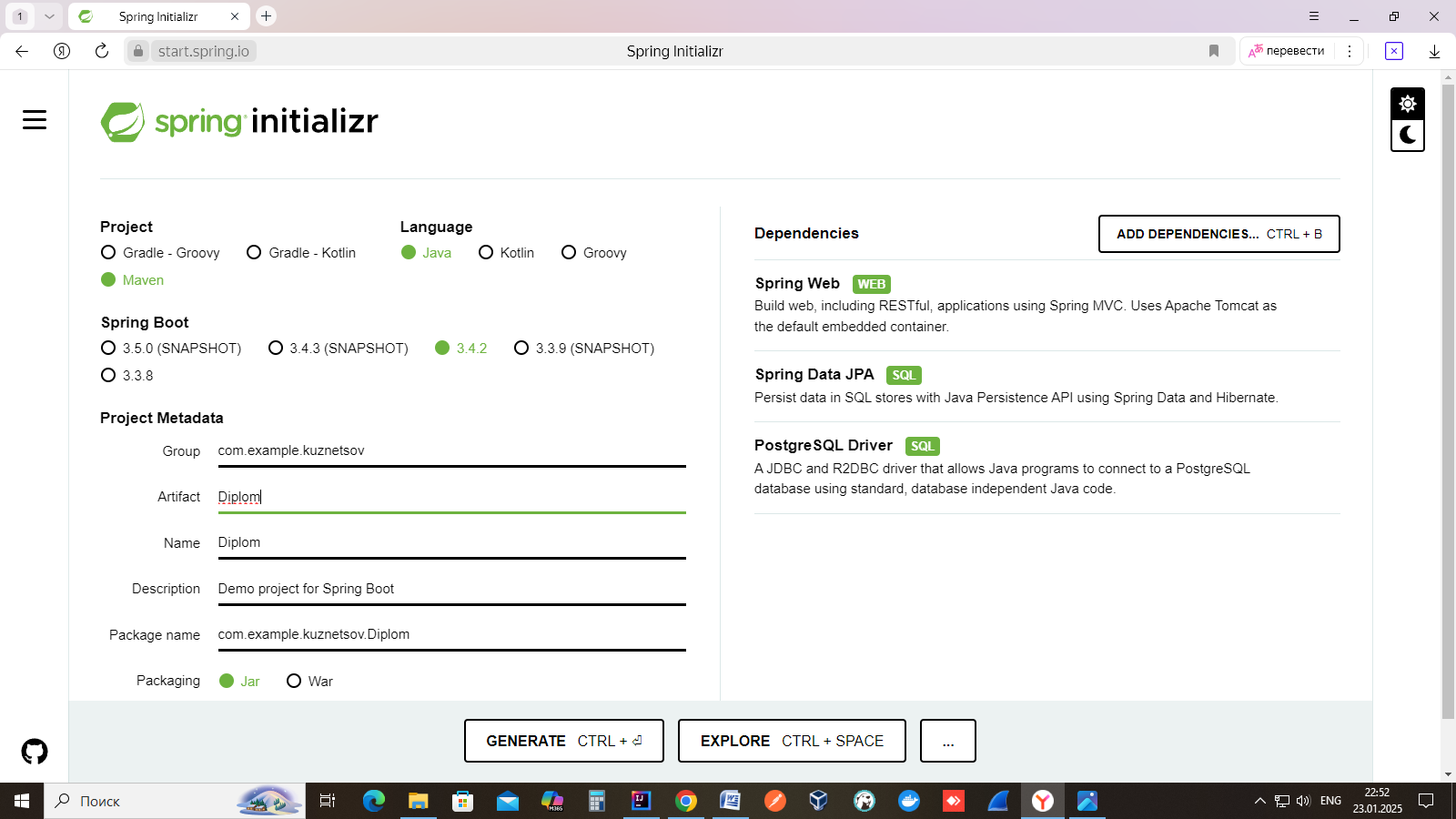
**Независимость от конкретного JPA-провайдера**. Spring Data JPA обеспечивает непрерывность работы между различными JPA-провайдерами, такими как Hibernate или EclipseLink, из-за своего абстрактного подхода к реализации.

**Удобство настройки и интеграции**. Spring Data JPA удобна в настройке благодаря поддержке конфигураций как на основе XML, так и на основе Java, что позволяет с лёгкостью интегрировать фреймворк в любую рабочую среду.

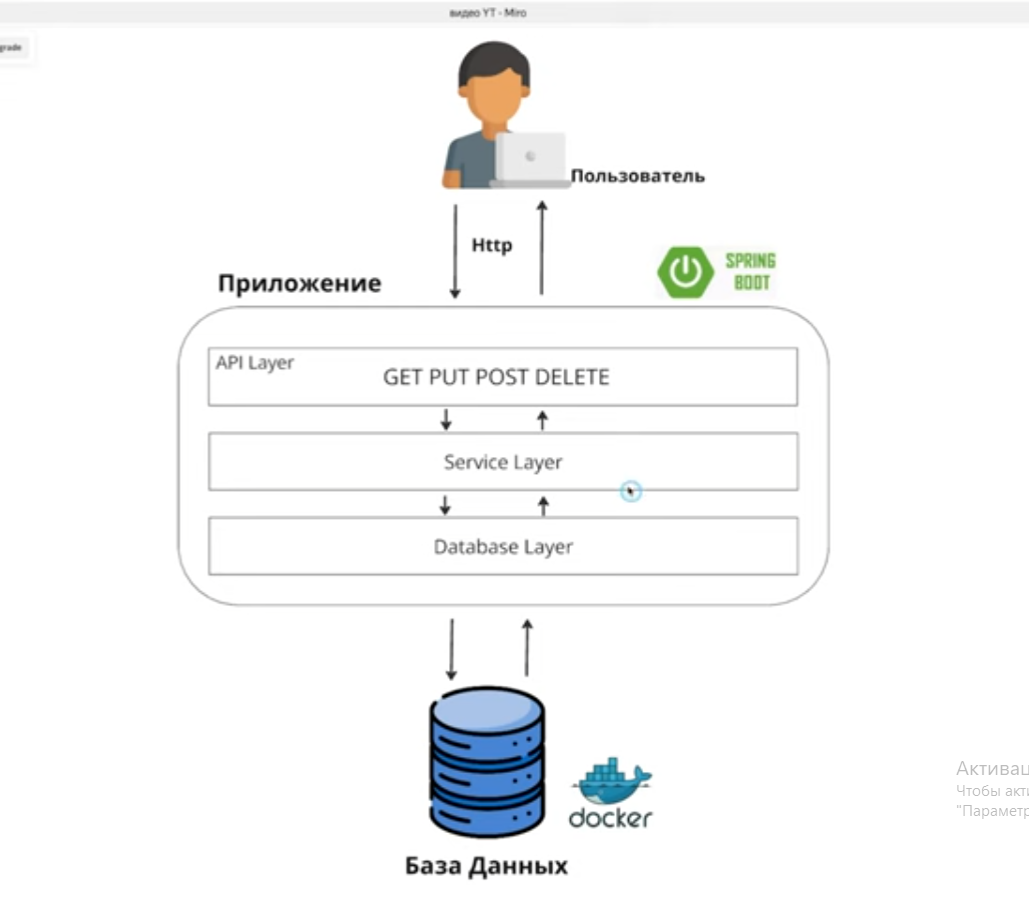
**PostgreSQL Driver SQL**A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

**Зависимость PostgreSQL Driver** **позволяет приложениям Java взаимодействовать с базами данных PostgreSQL**.

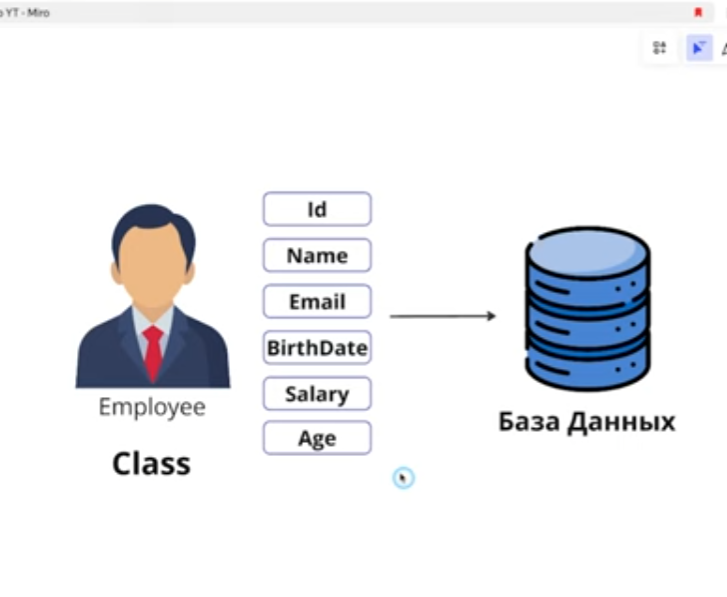
Драйвер преобразует команды из программы Java в запросы, связанные с базой данных, и обеспечивает открытие сокетных соединений от приложения к базе данных и отправку SQL-запросов.



Далее нажимаем Generate и получаем Demo файл. Далее разархивируем его в папку с нашим приложением.



Задачи которые будет выполнять наше приложение: получение, добавление сотрудника, изменение данных сотрудника (email, заработная плата), удаление сотрудника,. И все это будет связано с базой данных Postgresql. Которую мы поднимем с помощью Docker. Пользователь отправляет http запросы, наше приложение их обрабатывает.



Наш сотрудник будет иметь 6 полей указанных на рисунке, а пользователь, например сотрудник отдела кадров будет передавать их при помощи нашего приложения в базу данных.

Для начала создаем package Employees, а в нем класс Employee:

package com.example.kuznetsov.Diplom.employees;

import jakarta.persistence.\*;

import java.time.LocalDate;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@SequenceGenerator(name = "employee\_sequence",

sequenceName = "employee\_sequence",

allocationSize = 1)

@GeneratedValue(strategy = GenerationType.SEQUENCE,

generator = "employee\_sequence")

private Long id;

private String name;

private String email;

private LocalDate birthDate;

private Integer salary;

private Integer age;

public Employee(Long id, String name, String email, LocalDate birthDate, Integer salary, Integer age) {

this.id = id;

this.name = name;

this.email = email;

this.birthDate = birthDate;

this.salary = salary;

this.age = age;

}

public Long getId() {

return id;

}

public String getName() {

return name;

}

public String getEmail() {

return email;

}

public LocalDate getBirthDate() {

return birthDate;

}

public Integer getSalary() {

return salary;

}

public Integer getAge() {

return age;

}

public Employee() {

}

public void setEmail(String email) {

this.email = email;

}

public void setSalary(Integer salary) {

this.salary = salary;

}

}

**В данном классе созданы поля, Getters, Setters, Конструктор, пустой конструктор для работы с Hibernetes также аннотации для работы Spring.**

Далее создаем package employees в нем class EmployeeController. Данный класс служит для обработки запросов. Принимает в себя данные из HTTP запросов. И передает данные слою бизнес логики.

package com.example.kuznetsov.Diplom.controllers;

import com.example.kuznetsov.Diplom.employees.Employee;

import com.example.kuznetsov.Diplom.services.EmployeeService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

public class EmployeeController {

private final EmployeeService employeeService;

@Autowired

public EmployeeController(EmployeeService employeeService) {

this.employeeService = employeeService;

}

@GetMapping

public List<Employee> getAllEmployee() {

return employeeService.getAllEmployees();

}

@PostMapping

public Employee createEmployee(

@RequestBody Employee employee) {

return employeeService.createEmployee(employee);

}

@DeleteMapping("/{employeeId}")

public void deleteEmployee(

@PathVariable("employeeId") Long id) {

employeeService.deleteEmployee(id);

}

@PutMapping("/{employeeId}")

public void updateEmployee(

@PathVariable("employeeId") Long id,

@RequestParam(value = "email", required = false) String email,

@RequestParam(value = "salary", required = false) Integer salary) {

employeeService.updateEmployee(id,email,salary);

}

}

Далее создаем package services в нем class EmployeeServise. Здесь описана бизнес логика по работе с сотрудниками, базой данных и тд.

package com.example.kuznetsov.Diplom.services;

import com.example.kuznetsov.Diplom.employees.Employee;

import com.example.kuznetsov.Diplom.repositories.EmployeeRepository;

import jakarta.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class EmployeeService {

private final EmployeeRepository employeeRepository;

@Autowired

public EmployeeService(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

public Employee createEmployee(Employee employee) {

if (employee.getId() != null) {

throw new IllegalArgumentException("id must be empty");

}

if (employeeRepository.findByEmail(employee.getEmail())

.isPresent()) {

throw new IllegalArgumentException("Email already takes");

}

if (employee.getSalary() <= 5000) {

throw new IllegalArgumentException("Salary must be bigger than 5000");

}

return employeeRepository.save(employee);

}

public void deleteEmployee(Long id) {

if (employeeRepository.findById(id).isEmpty()){

throw new IllegalArgumentException("Employee not found by id=%s".formatted(id));

}

employeeRepository.deleteById(id);

}

@Transactional

public void updateEmployee(Long id, String email, Integer salary) {

{

var employee =employeeRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Employee not found by id=%s" .formatted(id)));

if (email != null

&& !email.isEmpty()

&& !email.equals(employee.getEmail()))

{

Optional<Employee> employeeOpt = employeeRepository.findByEmail(email);

if (employeeOpt.isPresent()){

throw new IllegalArgumentException("Email already taken");

}

employee.setEmail(email);

}

if (salary != null){

if (salary <= 5000){

throw new IllegalArgumentException("Salary must be bigger than 5000");

}

employee.setSalary(salary);

}

}

}

}

Далее создаем package repositories в нем interface EmployeeRepository. Он занимается связью с базой данных.

package com.example.kuznetsov.Diplom.repositories;

import com.example.kuznetsov.Diplom.employees.Employee;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

Optional<Employee> findByEmail(String email);

}

Также для того чтобы поднять базу данных с помощью Docker создаем файл

Docker-compose.yaml:

services:  
 db\_auth:  
 container\_name: db\_auth  
 ports:  
 - "5432:5432"  
 environment:  
 - POSTGRES\_PASSWORD=postgres  
 - POSTGRES\_USER=postgres  
 image: postgres:14.3-alpine

Также настроим файл application properties:

spring.application.name=Diplom  
  
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres  
spring.datasource.username=postgres  
spring.datasource.password=postgres  
spring.jpa.hibernate.ddl-auto=create-drop  
spring.jpa.show-sql=true  
server.error.include-message=always

Далее создадим class EmployeeConfig для инициализации дефолтных работников:

package com.example.kuznetsov.Diplom.configuration;

import com.example.kuznetsov.Diplom.employees.Employee;

import com.example.kuznetsov.Diplom.repositories.EmployeeRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import java.time.LocalDate;

import java.util.List;

@Configuration

public class EmployeeConfig {

@Bean

CommandLineRunner commandLineRunner(

EmployeeRepository employeeRepository

) {

return (args) -> {

var employeeList = List.of(new Employee(

null,

"Vasiliy",

"vasekku25@gmail.com",

LocalDate.of(1976, 11, 19),

90000,

48

),

new Employee(

null,

"Pavel",

"pavel@gmail.com",

LocalDate.of(1975, 4, 6),

80000,

49

)

);

employeeRepository.saveAll(employeeList);

};

}

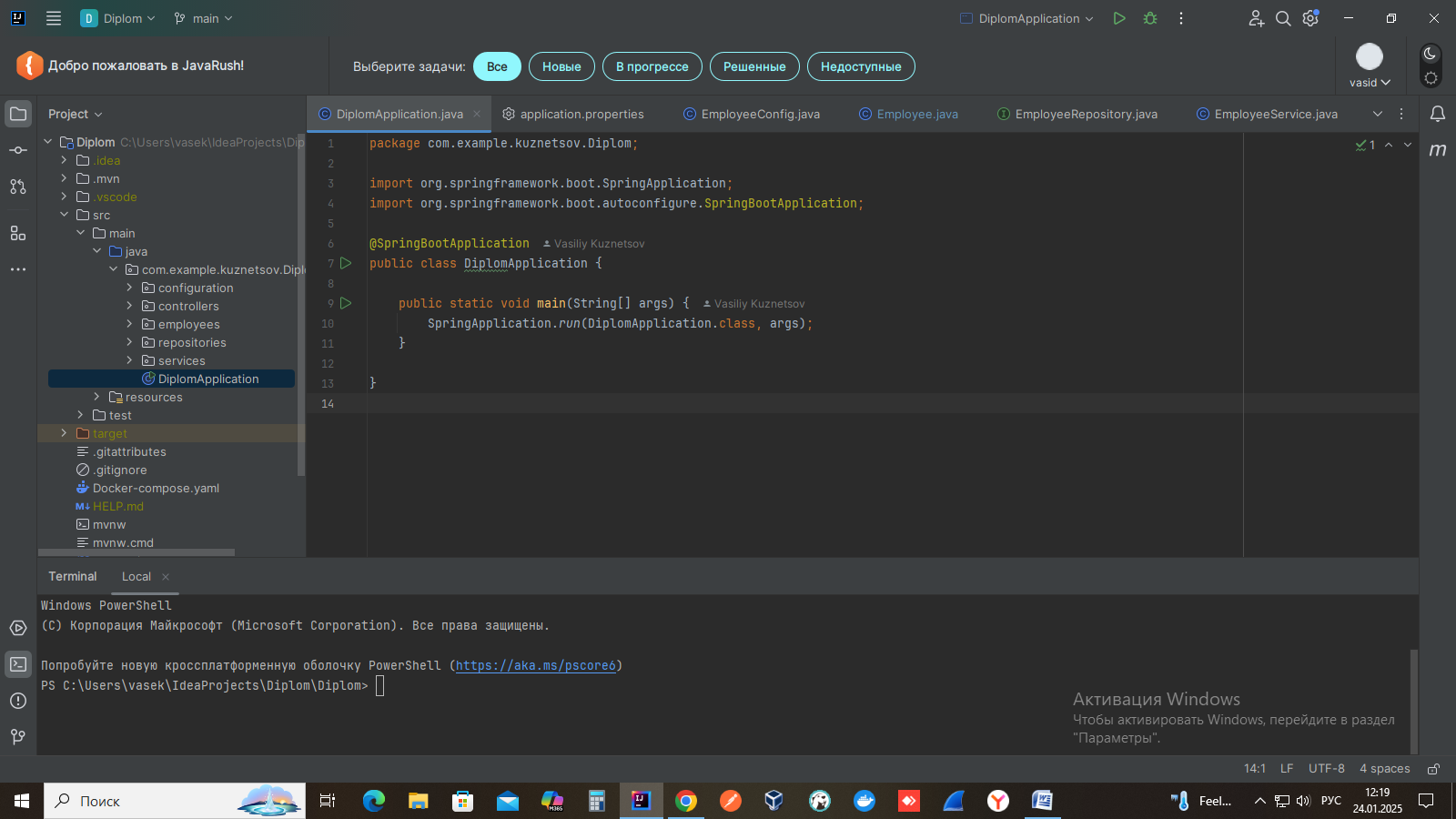
}

Id указывать не нужно , он автоматически будет подставляться в БД.

Проверить работу приложения будем с помощью программы Postman

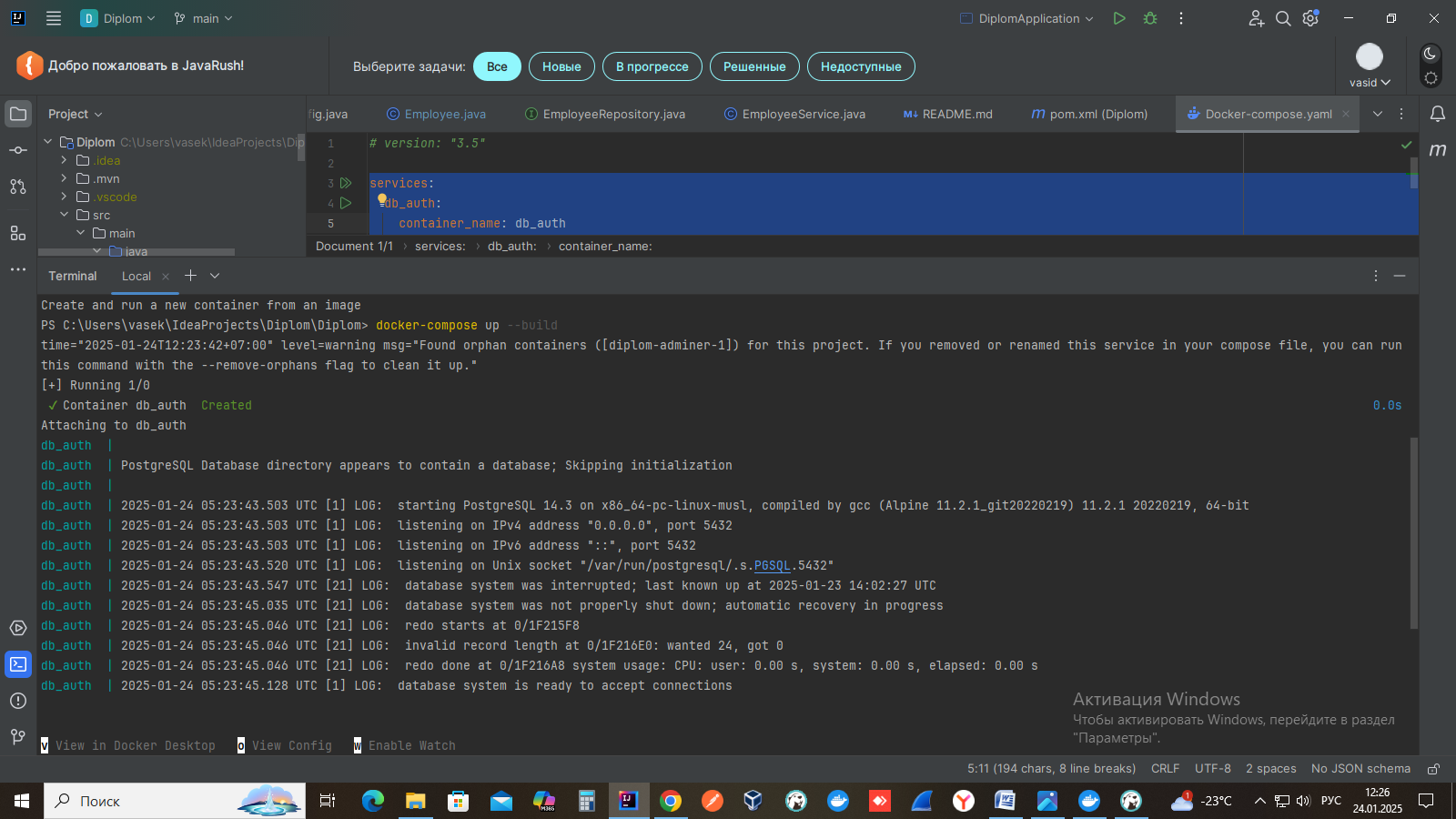
Базу данных откроем в DBeaver.

Приступим к запуску программы. Открываем InelijIde открываем нашу папку.



Далее запускаем Docker.

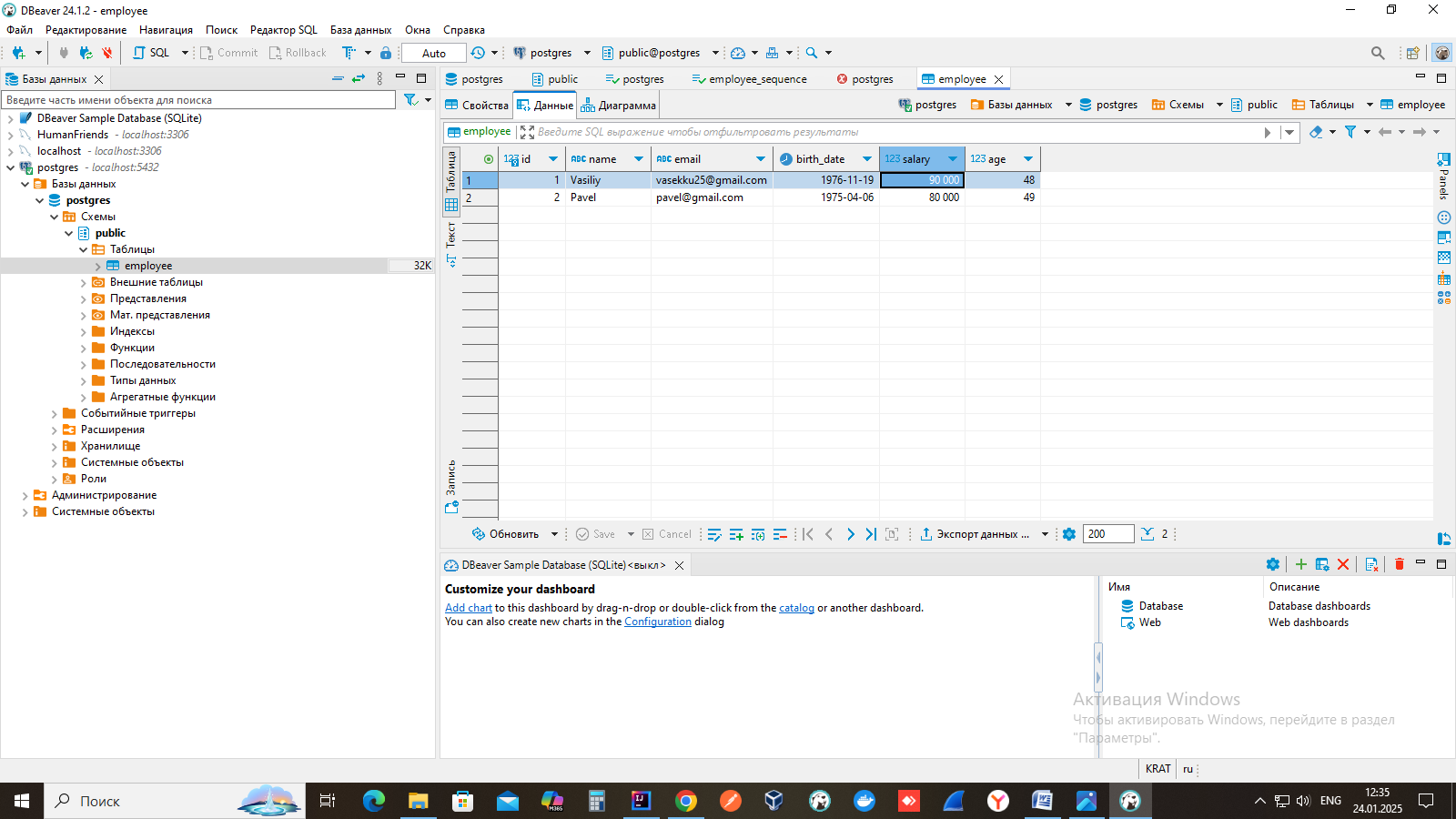
Затем в терминале прописываем docker-compose up –build, для сборки контейнера.



Docker-compose.yaml –p diplom up –d. Запускаем Docker-compose.yaml подключаемся к базе данных на порту 5432

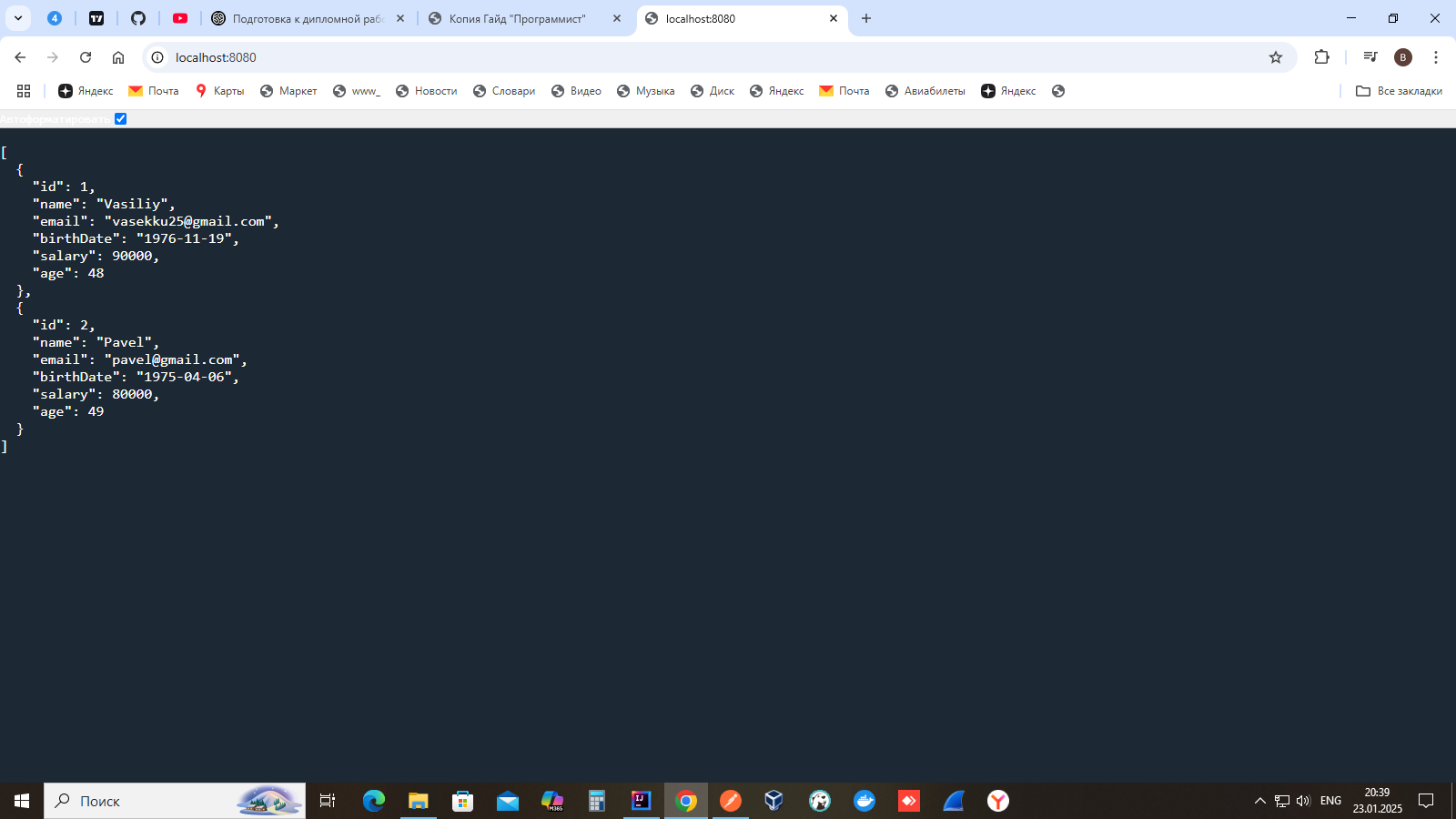
Стартуем наше приложение из Classa DiplomApplication

Приложение стартовало удачно мы это можем увидеть в Dbeaver



Наши дефолтные сотрудники появились в БД.

Также мы можем проверить на Localhost:8080 Json Объекты данных пользователей



Далее мы хотим добавить сотрудника в базу данных для этого мы отправим в Postman запрос Json

{

"name": "Bob",

"email": "Bob@gmail.com",

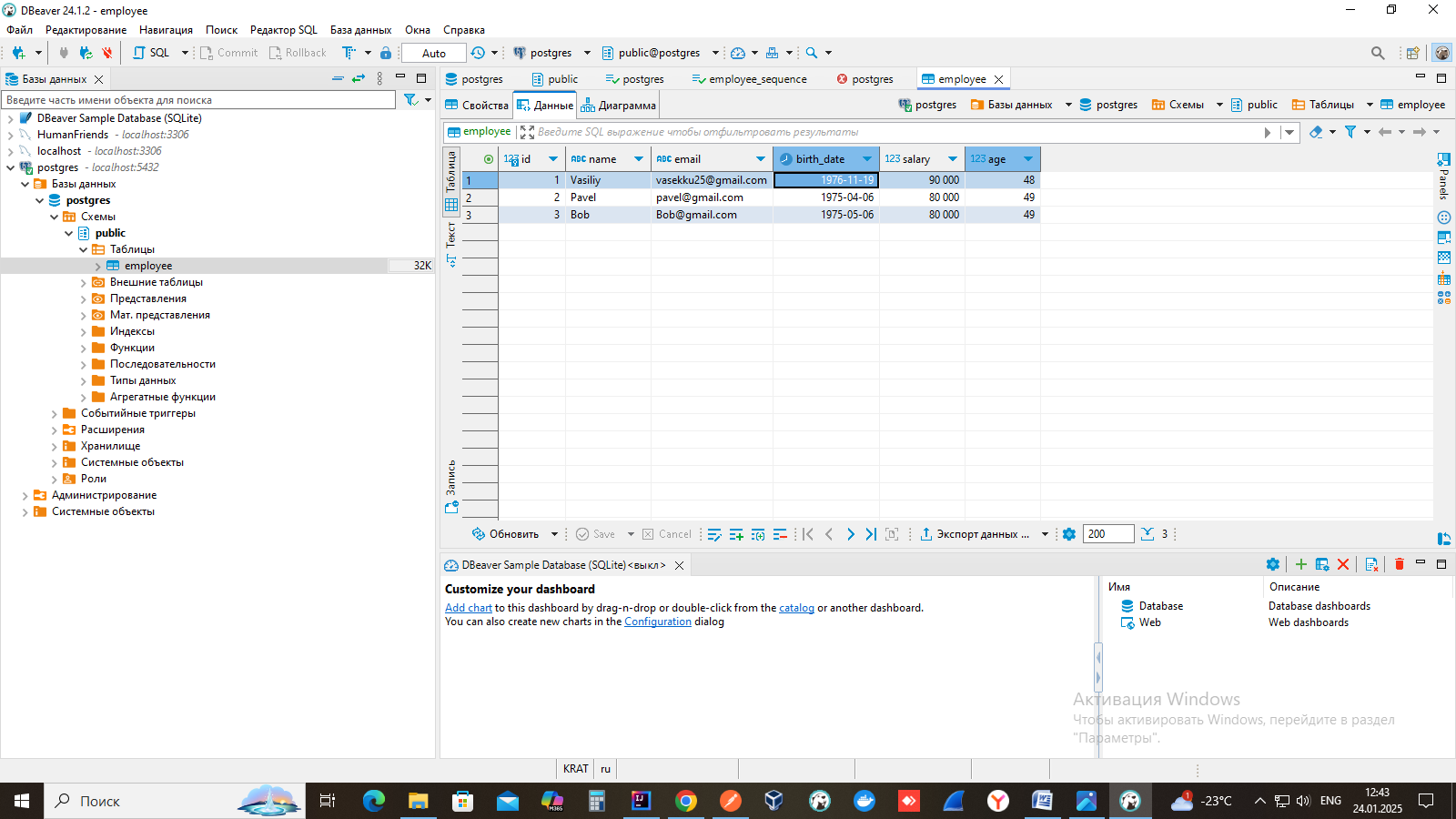
"birthDate": "1975-05-06",

"salary": 80000,

"age": 49

}

Видим , что сотрудник благополучно добавился в бд



public Employee createEmployee(Employee employee) {  
  
 if (employee.getId() != null) {  
 throw new IllegalArgumentException("id must be empty");  
 }  
  
 if (employeeRepository.findByEmail(employee.getEmail())  
 .isPresent()) {  
 throw new IllegalArgumentException("Email already takes");  
 }  
  
 if (employee.getSalary() <= 5000) {  
 throw new IllegalArgumentException("Salary must be bigger than 5000");  
  
 }  
  
 return employeeRepository.save(employee);  
}

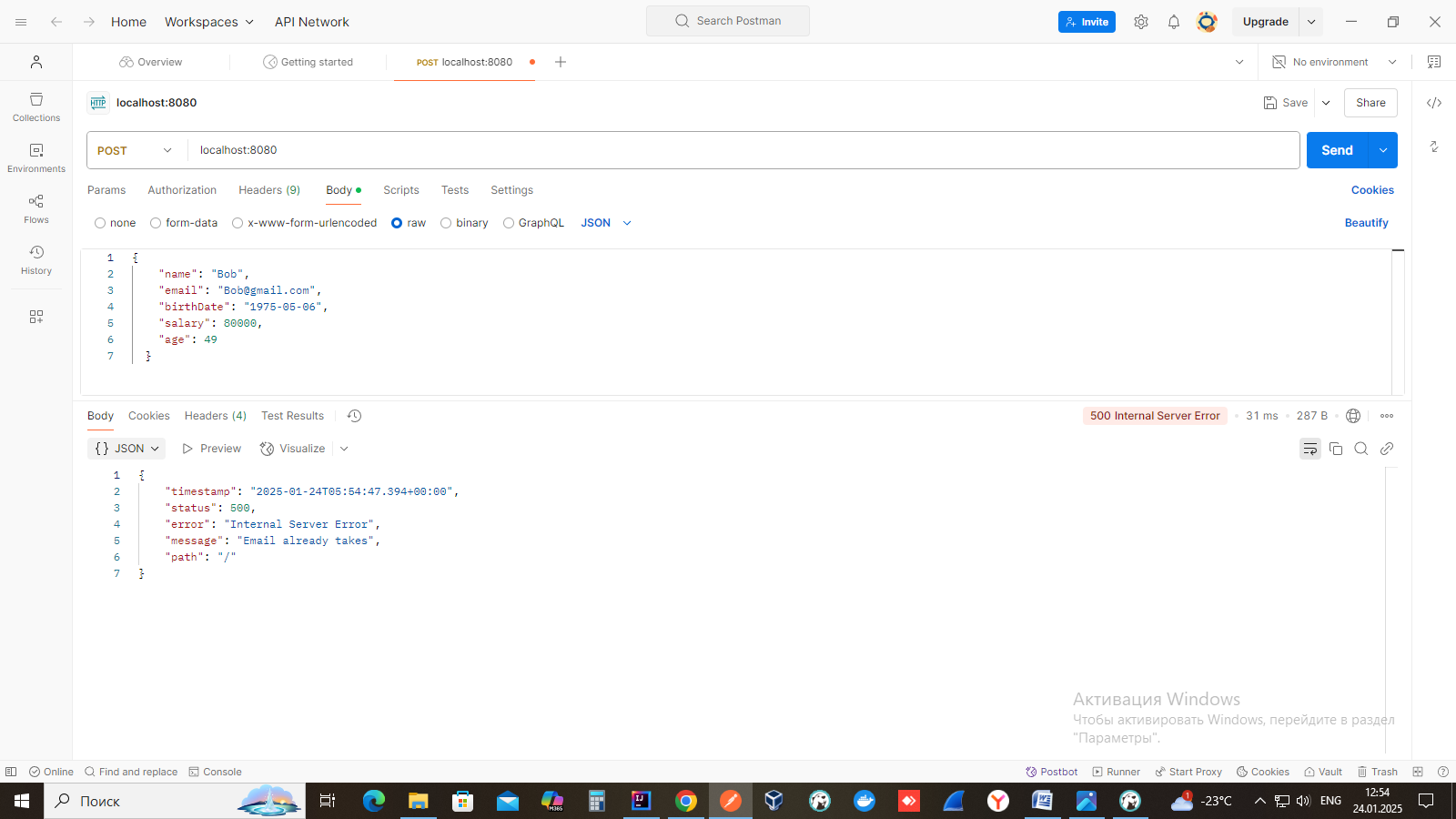
В данный метод я добавил несколько проверок.

1.если в запросе будет указан ID выброситься исключение “идентификатор должен быть пустым”

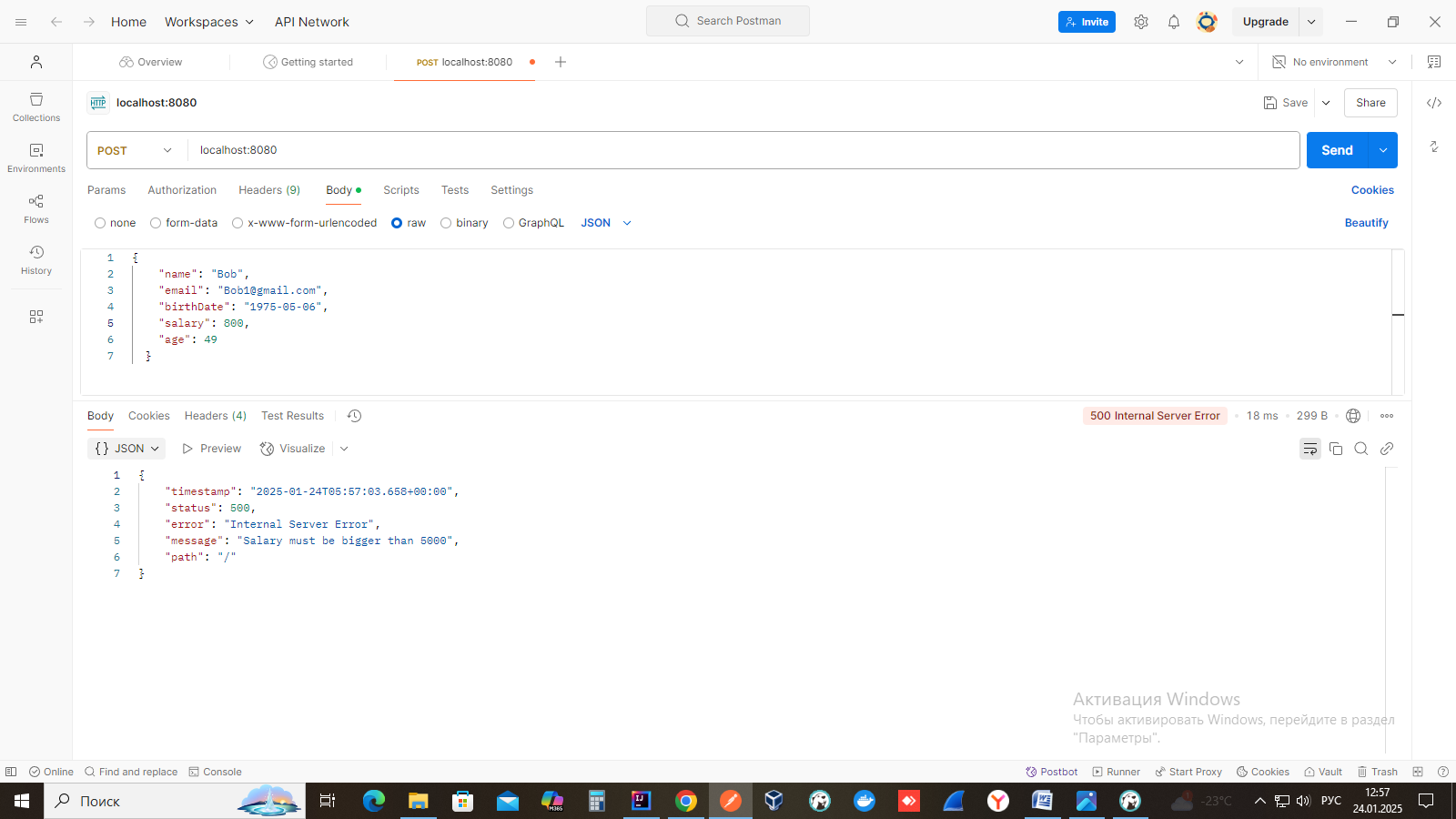
2. Если электронная почта будет повторяться также будет исключение.

3. Если например заработная плата будет меньше положенной например по законодательству также будет исключение, например в моем случае 5000р

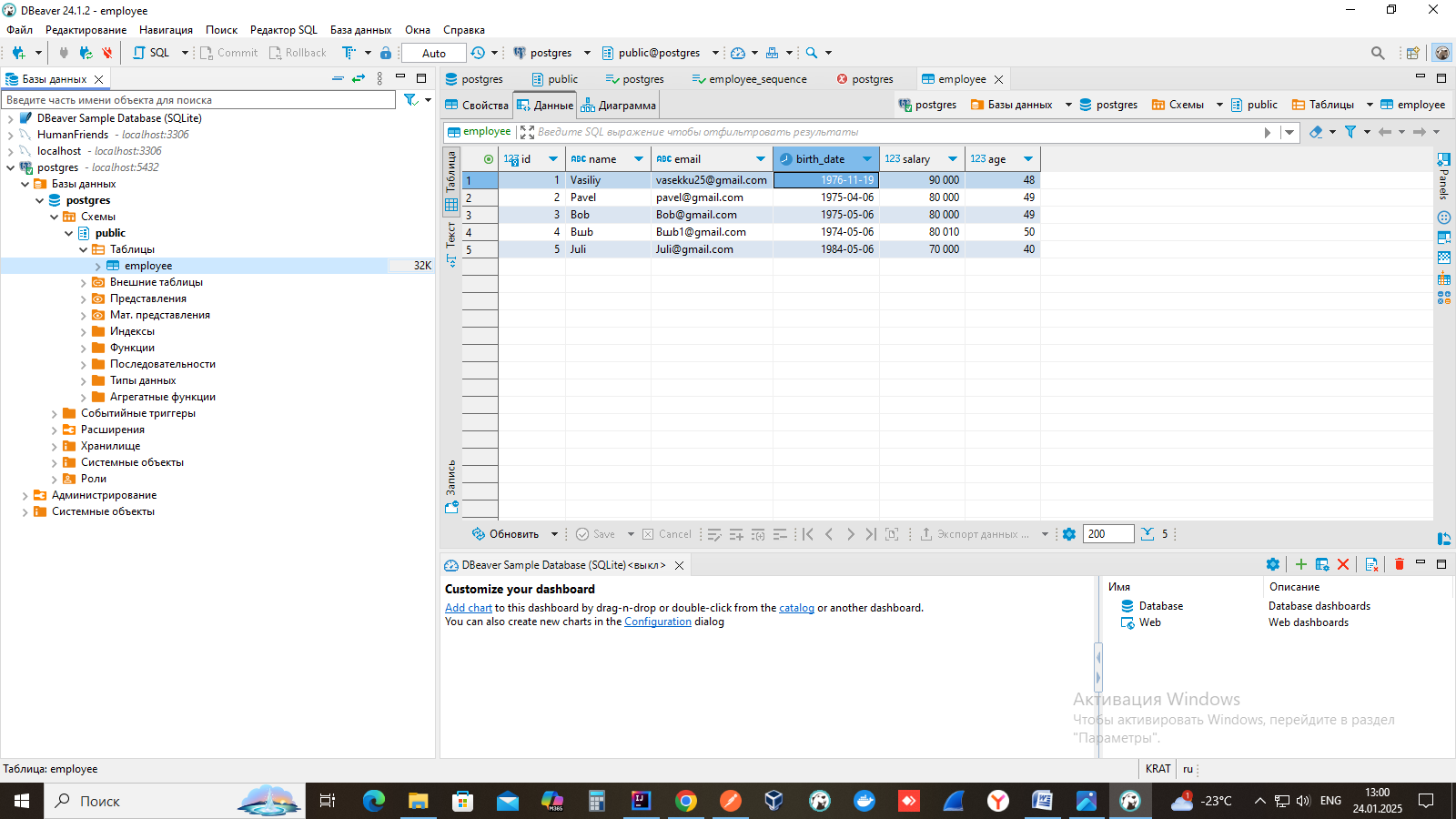
Пробуем завести сотрудника с той же почтой

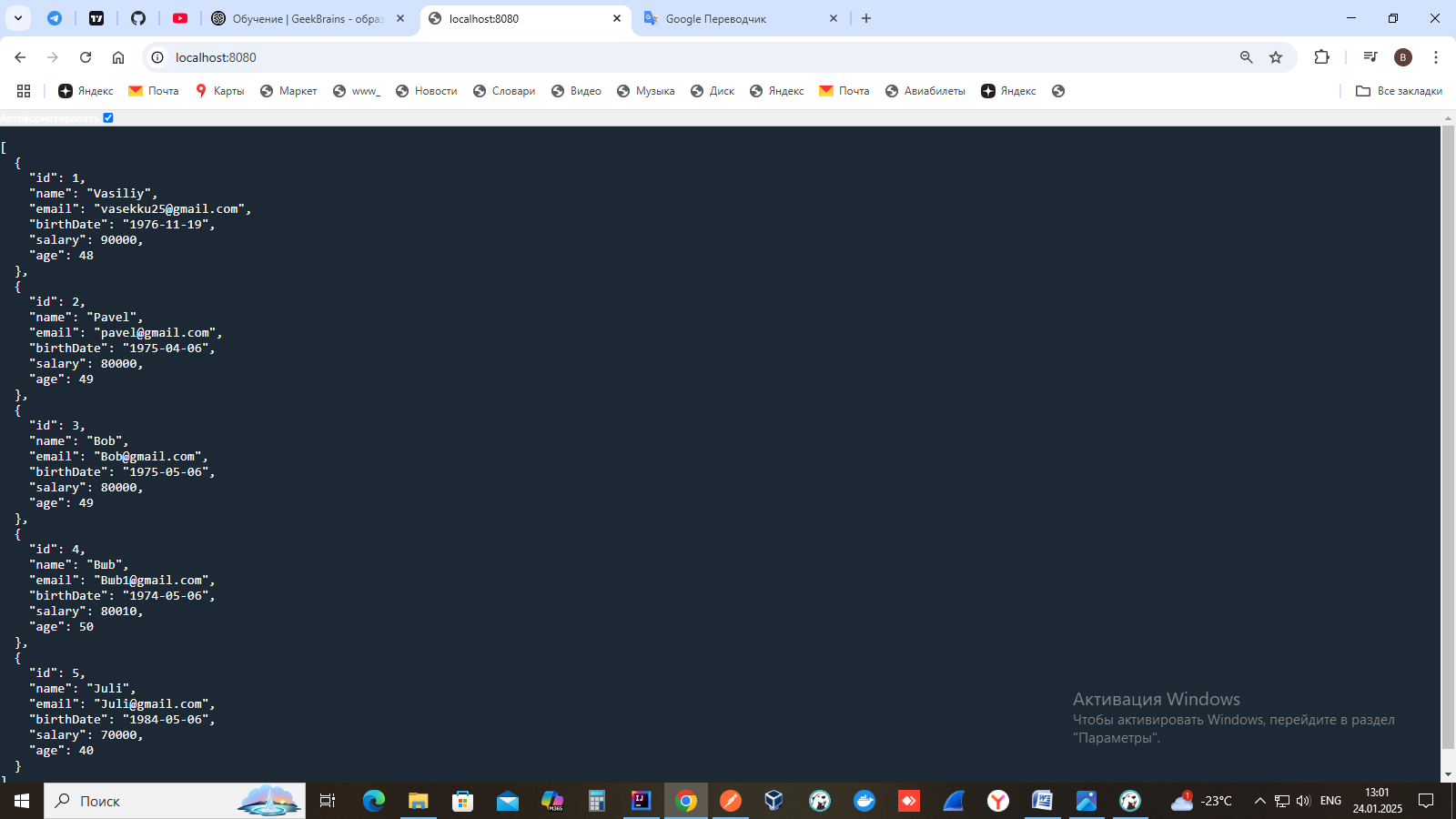


Пробуем завести сотрудника зарплатой менее 5000р.



Метод добавления сотрудников прекрасно работает добавим еще несколько человек



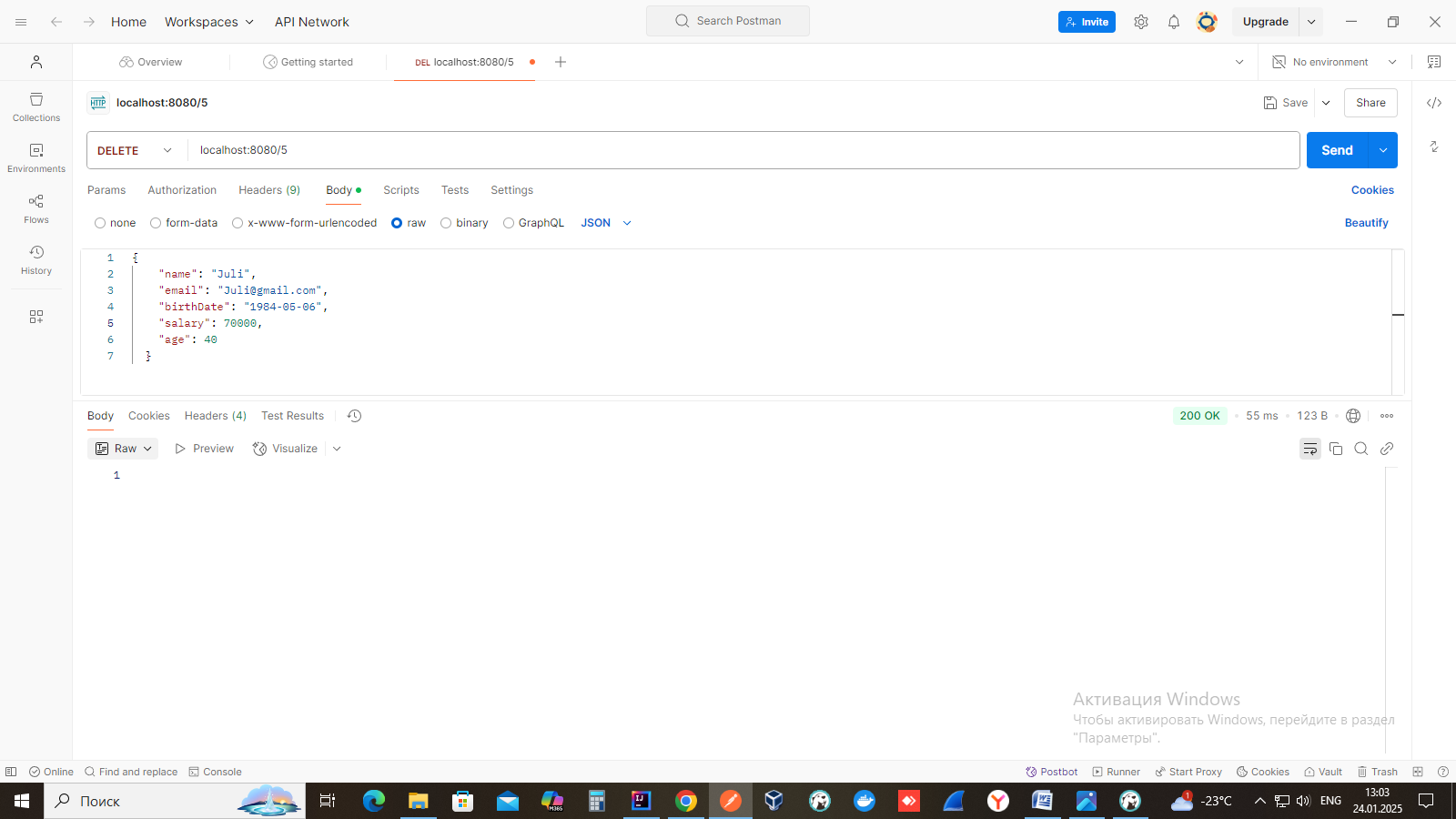


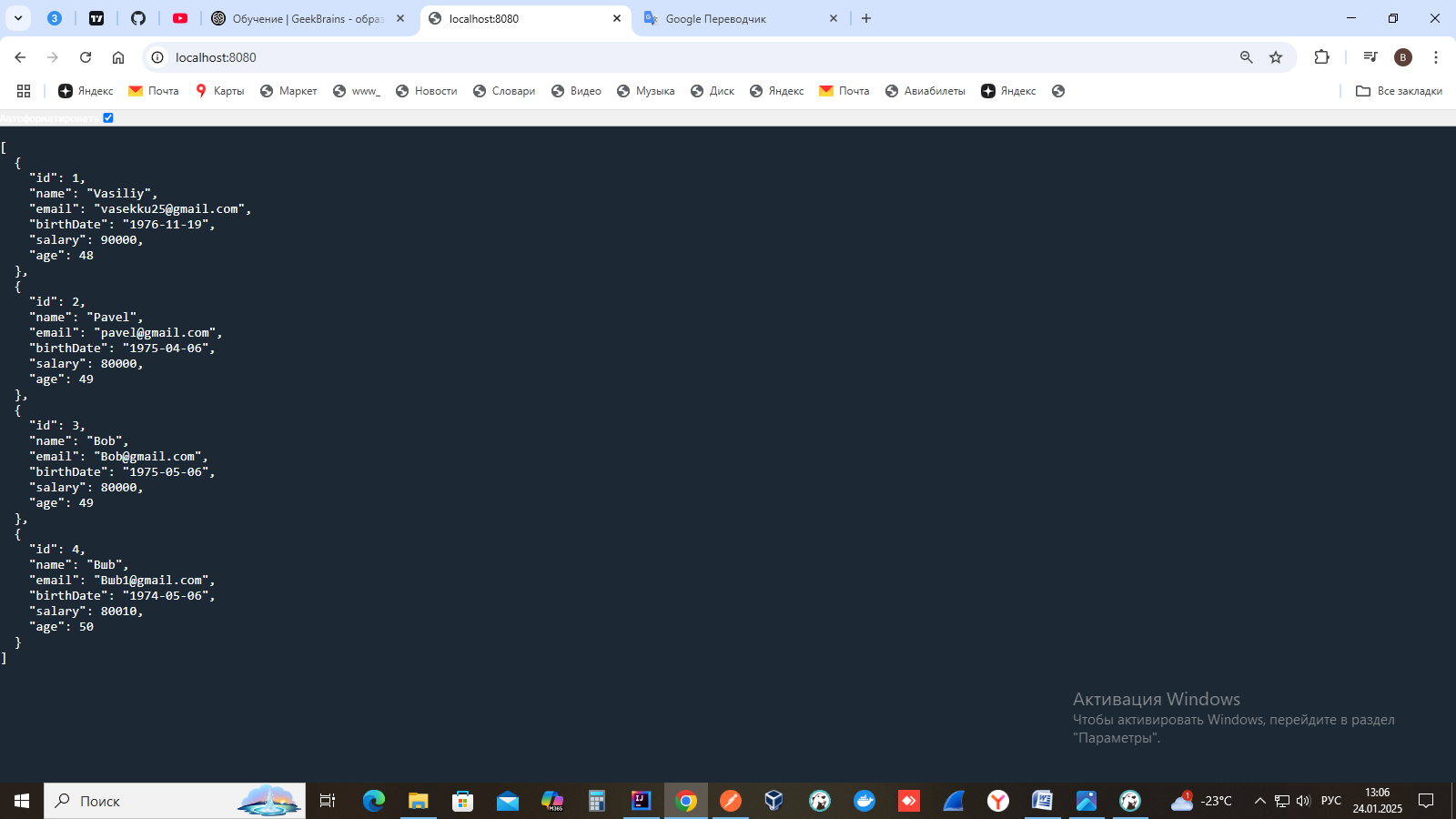
Далее рассмотрим метод удаления работника

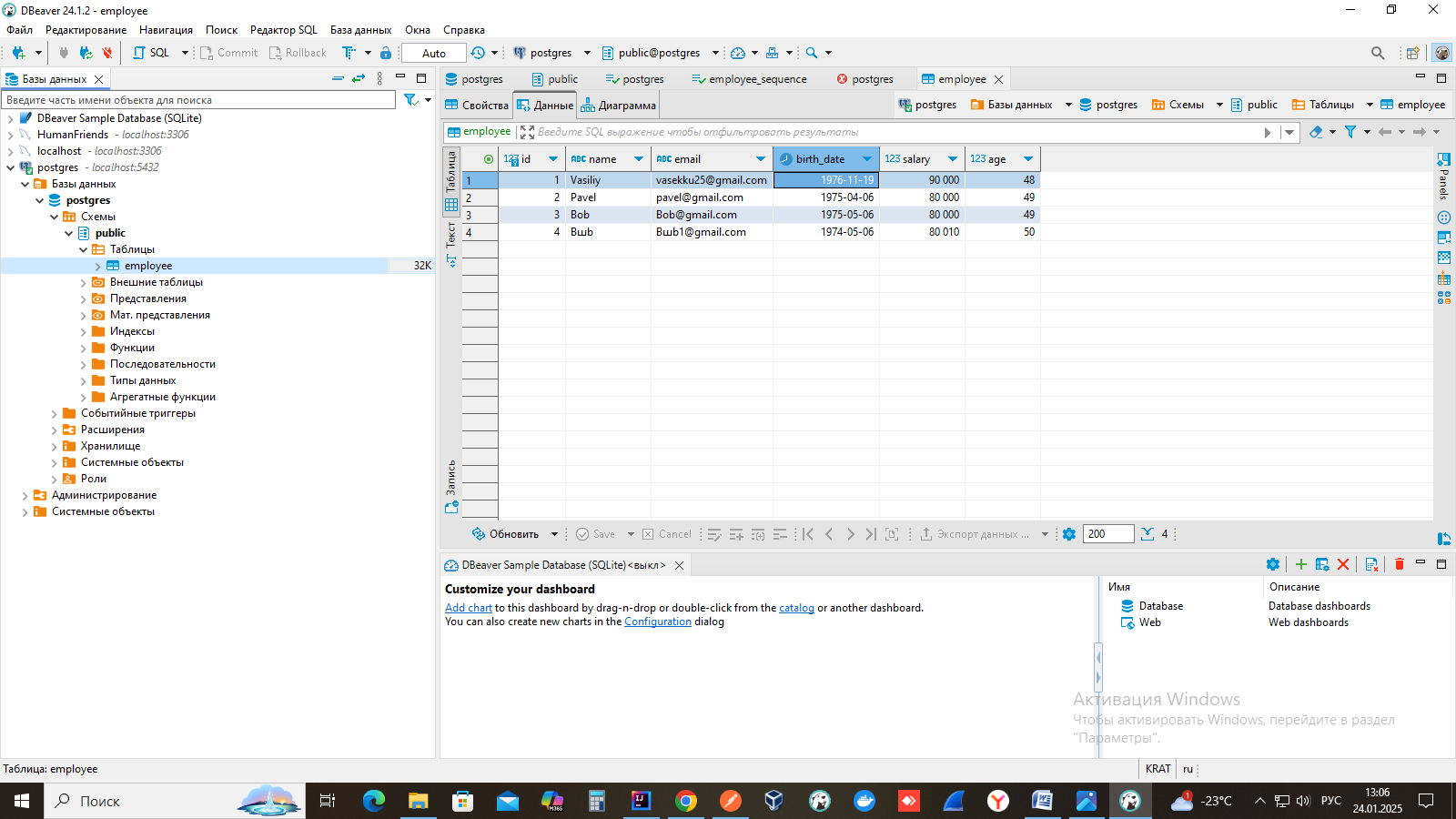
public void deleteEmployee(Long id) {  
 if (employeeRepository.findById(id).isEmpty()){  
 throw new IllegalArgumentException("Employee not found by id=%s".formatted(id));  
 }  
 employeeRepository.deleteById(id);  
}

Работник удаляется очень просто делаем запрос Delete localhost:8080/5.

Через слеш указываем IDсотрудника и он удаляется из БД







Далее у нас метод по обновлению данных о сотруднике, например Email и заработная плата.

@Transactional  
public void updateEmployee(Long id, String email, Integer salary) {  
 {  
 var employee =employeeRepository.findById(id)  
 .orElseThrow(() -> new IllegalArgumentException("Employee not found by id=%s"  
 .formatted(id))  
 );  
 if (email != null  
 && !email.isEmpty()  
 && !email.equals(employee.getEmail()))  
 {  
 Optional<Employee> employeeOpt = employeeRepository.findByEmail(email);  
 if (employeeOpt.isPresent()){  
 throw new IllegalArgumentException("Email already taken");  
  
 }  
  
 employee.setEmail(email);  
 }  
 if (salary != null){  
 if (salary <= 5000){  
 throw new IllegalArgumentException("Salary must be bigger than 5000");  
  
 }  
 employee.setSalary(salary);  
 }  
  
 }  
}

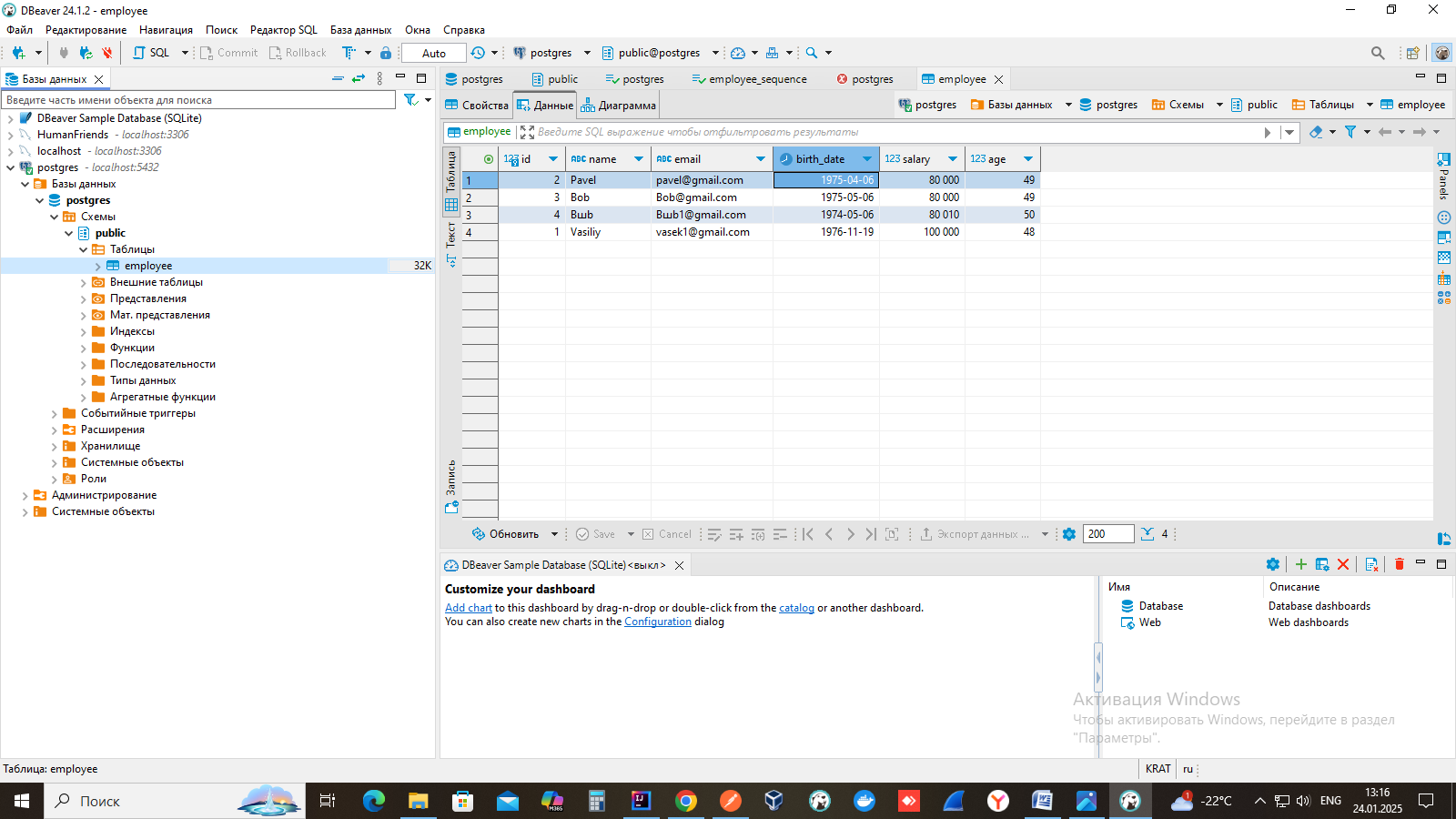
Тут у нас также присутствует несколько проверок они схожи с предыдущим методом.

Здесь при помощи запросов мы можем изменить зарплату и email одновременно или по отдельности.

Запросы выглядят следующим образом

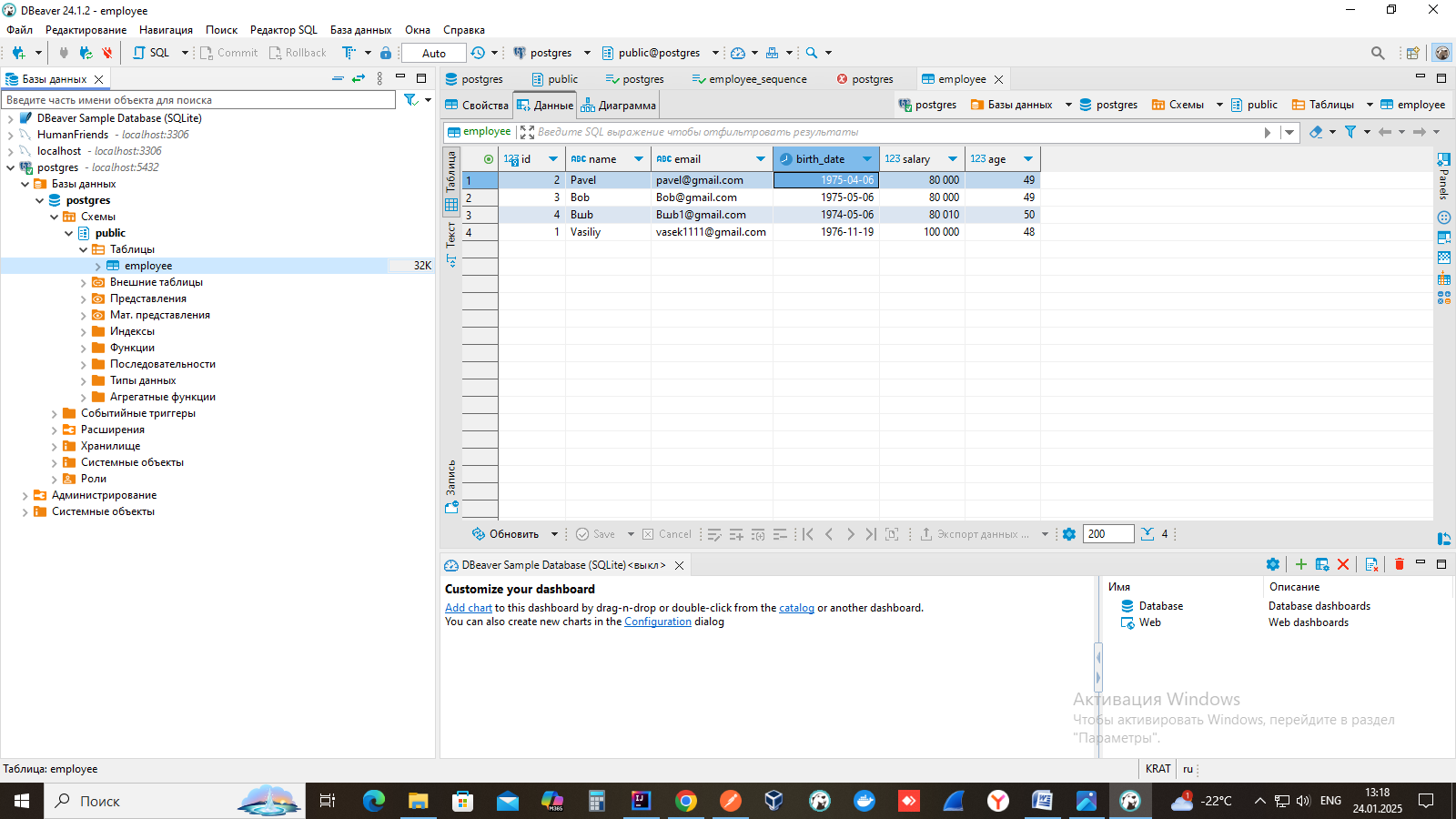
Put localhost:8080/1?email=vasek1@gmail.com&salary=100000

Тут мы меняем сотруднику с ID1 почту и зарплату



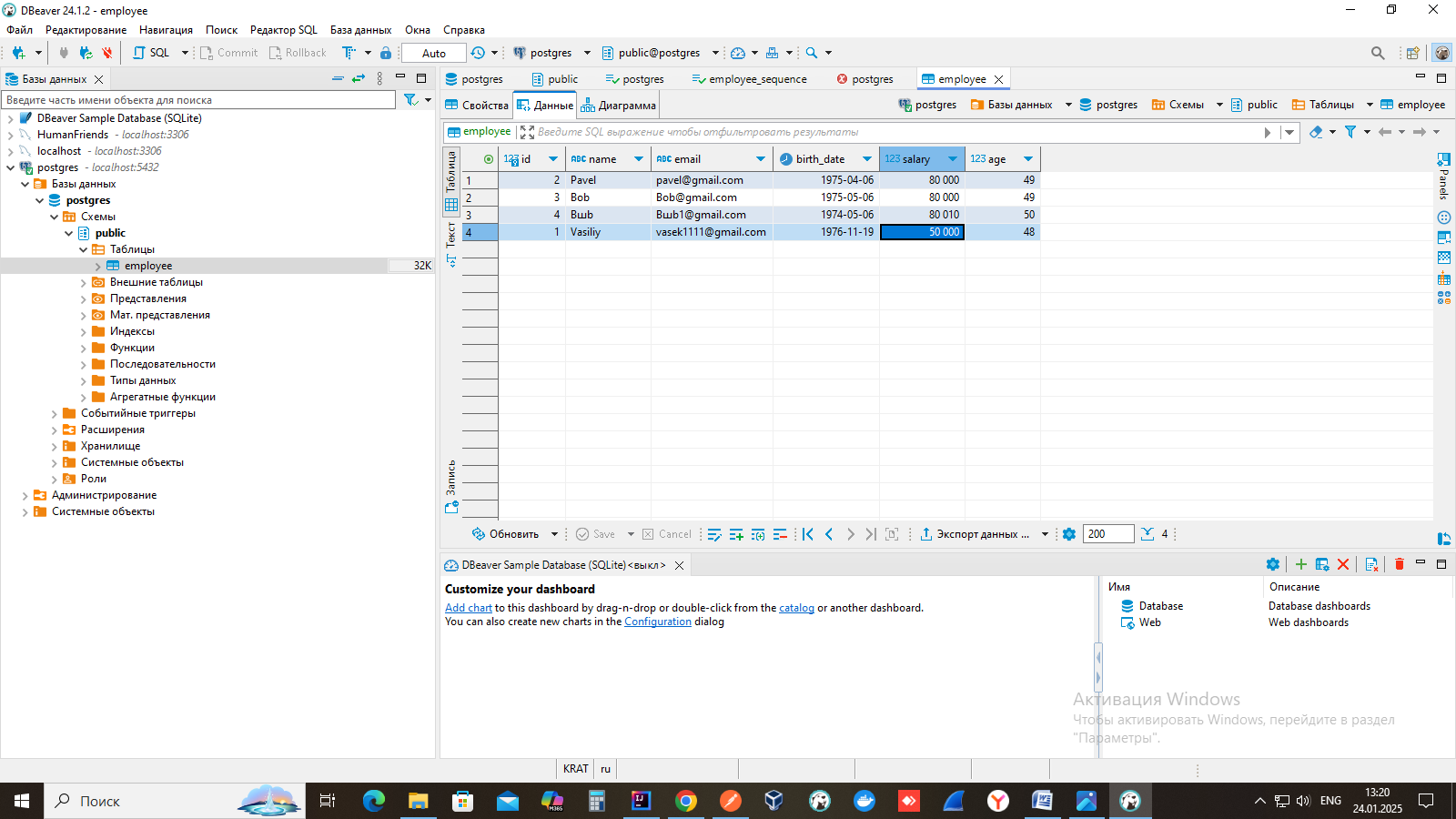
**Просто email**

Put localhost:8080/1?email=vasek1111@gmail.com



**Просто заработную плату**

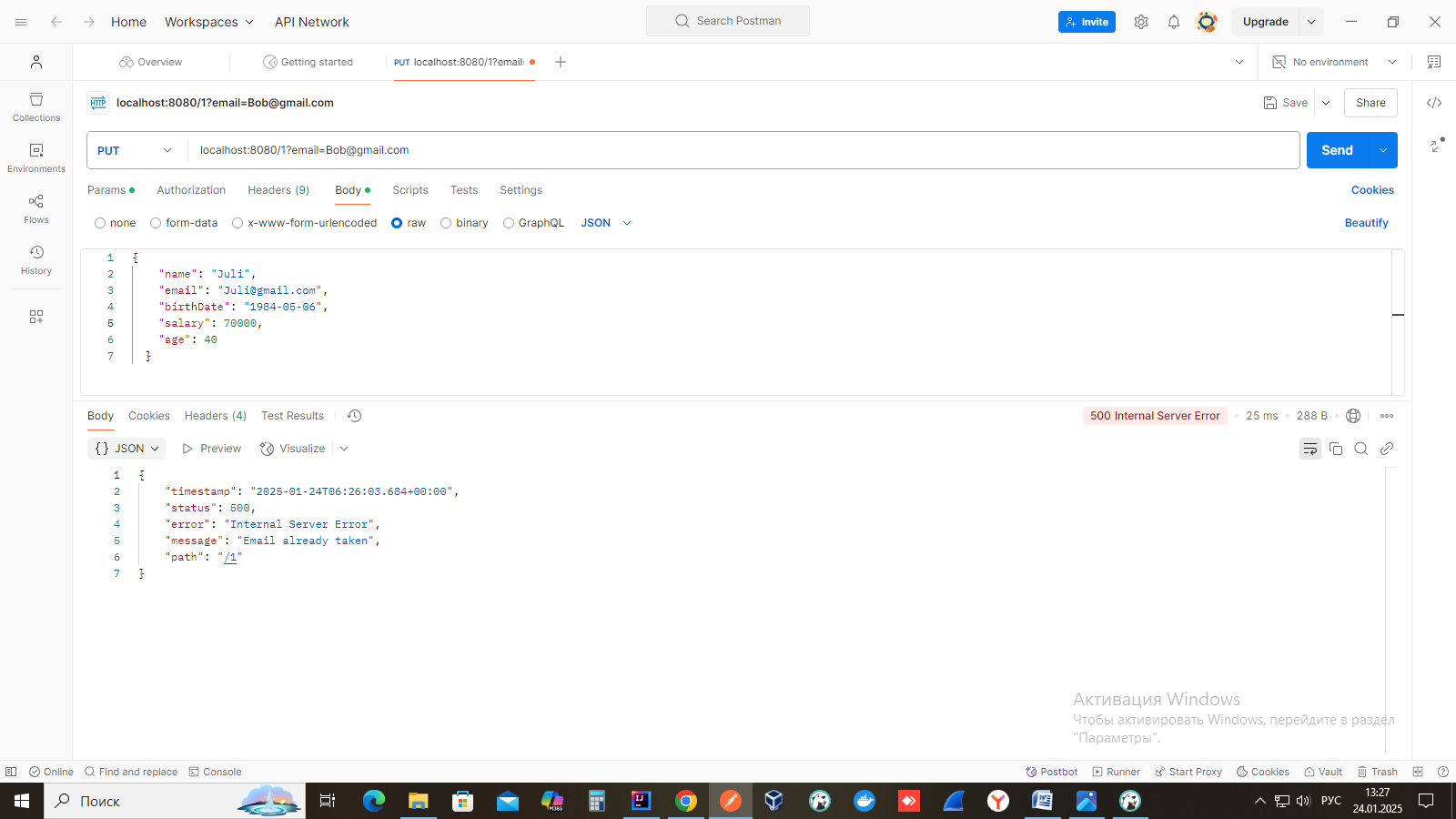
Put localhost:8080/1?salary=50000



**Если меняем зарплату ниже минимальной получаем исключение**



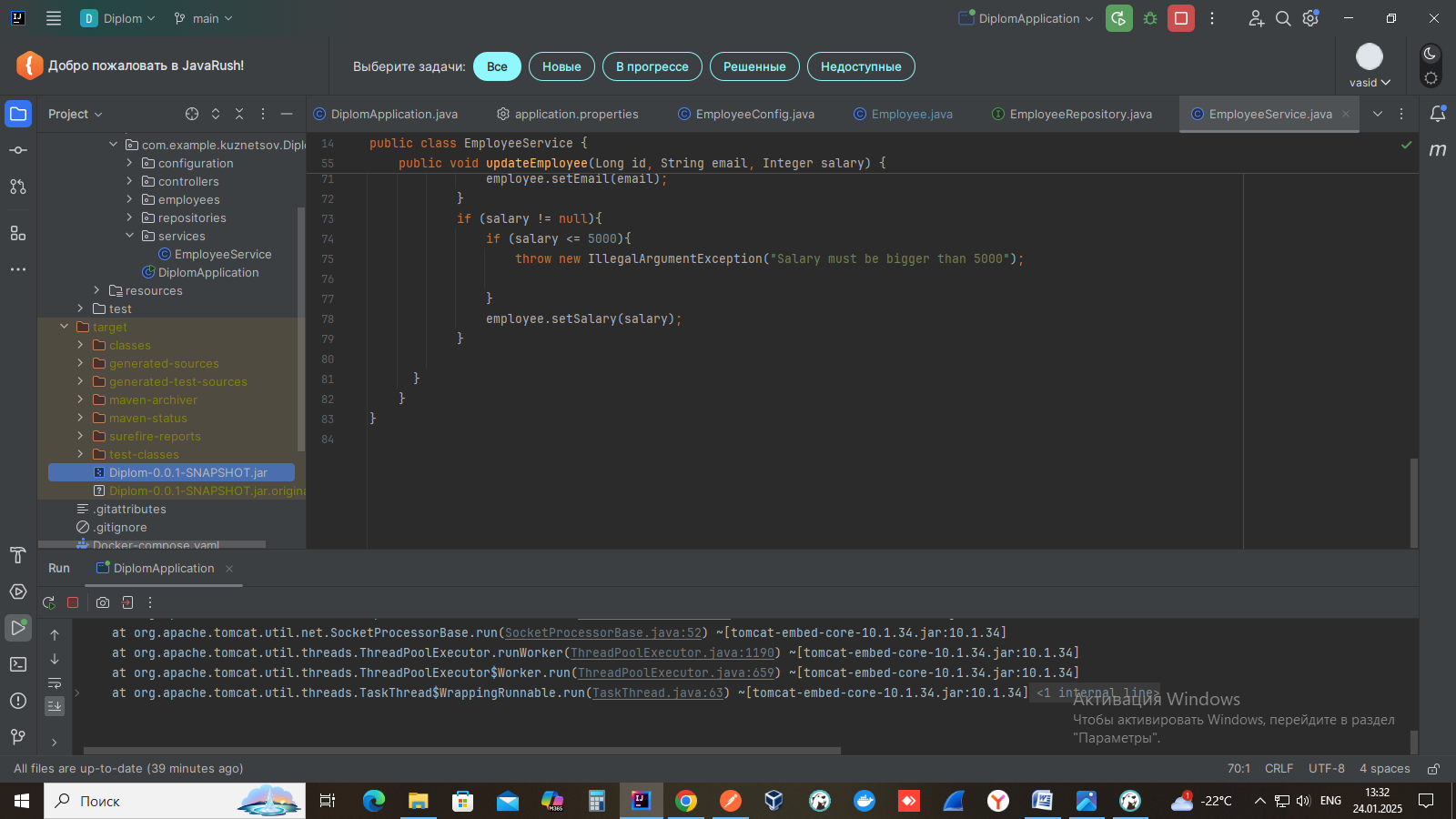
**Если мы вводим пустой или кем то занятый email также получаем исключение**



**Наше приложение работает и теперь мы соберем его в jar файл, чтобы можно було отправить и запустить на каких то других серверах через console. Заходим в IDE в Maven делаем clean очищаем папку target. Далее нажимаем package и собираем файл.**

**У нас он называется:**





**Данный файл мы можем запустить с помощью команды**

**Java -jar Diplom-0.0.1-SNAPSHOT.jar**

Глава 4. Заключение

**В заключении к диплому о приложении по учёту сотрудников на предприятии с использованием Spring Boot, Spring Web и базы данных PostgreSQL можно написать следующее:**

В ходе выполнения дипломного проекта были рассмотрены и применены теоретические и практические аспекты разработки веб-приложения для упраления сотрудниками на предприятиях. Теоретический анализ охватывал современные подходы к созданию REST API, методы работы с базами данных PostgreSQL, а также преимущества и особенности контейнеризации с Docker, что упрощает развертывание и поддержку масштабируемости приложения. Практическая часть проекта подтвердила эффективность для построения гибкого и масштабируемого решения, которое обеспечивает надежное и удобное взаимодействие с пользователями и системами.

Разработанное приложение позволяет обрабатывать данные о сотрудниках и выполнять различные операции с каждой записью в базе данных. Для этого были использованы следующие технологии:

**Spring Boot**. Фреймворк упрощает создание автономных приложений производственного уровня, которые можно легко развернуть и запустить с минимальной конфигурацией.

**PostgreSQL**. Для интеграции с базой данных были добавлены зависимости spring-boot-starter-jdbc и postgresql, а также указаны параметры подключения (логин, пароль и т. п.)

В приложении реализованы такие функции, как добавление новых сотрудников в базу данных, обновление существующих сотрудников, удаление сотрудников, получение данных обо всех сотрудниках в виде таблицы.

Таким образом, разработанный проект демонстрирует, как Spring Boot и другие технологии могут быть использованы для создания эффективных систем управления персоналом.

Любое приложение не совершенно и в него можно добавлять новые поля и методы . например место жительства, семейное положение и тд. Можно произвести градацию сотрудников(Начальник, водитель, слесарь и тд)

**Для совершенствования приложения можно добавить следующие методы**:

* **Использовать Spring Boot Actuator**. Он предоставляет множество конечных точек для мониторинга приложения и управления им, что поможет быстро выявлять и устранять проблемы.
* **Поддерживать актуальность зависимостей**. Регулярное обновление зависимостей обеспечит доступ к новейшим функциям и исправлениям ошибок.
* **Использовать Profiler**. Встроенное профилирование приложения поможет выявить узкие места в производительности и утечки памяти.
* **Использовать поддержку тестирования Spring Boot**. Она позволяет быстро и легко писать и запускать тесты для приложения.
* **Использовать Spring Boot Security** для авторизации администраторов

# **Список используемой литературы**

* *Бондаренко, А. В. Программирование на Java. Полное руководство. — Москва: Питер, 2022.*
* *Виноградова, И. П., Смирнов, Ю. С. Современные базы данных: принципы и подходы. — Санкт-Петербург: БХВ-Петербург, 2020.*
* *Зенкевич, И. Л. Разработка RESTful API. Руководство по проектированию и реализации. — Москва: ДМК Пресс, 2019.*
* *Иванов, С. А. Docker и контейнеризация: практическое руководство. — Москва: ДМК Пресс, 2020.*
* *Кулагин, А. П., Петров, Н. И. Основы работы с PostgreSQL: теория и практика. — Екатеринбург: У-Фактория, 2021.*
* *Ричардсон, Крис. Микросервисы: разработка и развертывание. — Москва: Питер, 2018.*
* *Саммерфилд, М. Программирование на Java: полное руководство. — Москва: Лаборатория знаний, 2021.*
* *Фримен, Э., Робсон, Э. Spring в действии. — Москва: Эксмо, 2020.*
* *Эванс, Э. Domain-Driven Design: Сложность в сердце ПО. — Санкт-Петербург: Питер, 2021.*
* *Кэти Сьерра и Берт Бейтс. Изучаем JAVA 2-е издание-Москва 2022г*
* *Джошуа Блох. JAVA эффективное програмирование третье издание. Москва Санкт Петербург 2022г*
* *Роберт Мартин. Чистый код Санкт-Петербург,Москва,Минск 2024г*

тербург

*Интернет-ресурсы*

* *Docker Документация. Docker Overview. — URL:* [*https://docs.docker.com/get-started/overview/*](https://docs.docker.com/get-started/overview/)
* *Oracle Java Документация. — URL:* [*https://docs.oracle.com/en/java/*](https://docs.oracle.com/en/java/)
* *Spring Framework Документация. — URL:* [*https://docs.spring.io/spring-framework/docs/current/reference/html/*](https://docs.spring.io/spring-framework/docs/current/reference/html/)
* *Система вопросов и ответов о программировании, разработанная Джоэлем Спольски и Джеффом Этвудом в 2008 году. — URL:**https://stackoverflow.com/*
* *Платформа GeekBrains*