Санкт-Петербургский Политехнический Университет

Институт компьютерных наук и технологий

«Высшая школа программной инженерии»

**Отчёт по лабораторной работе**

по дисциплине: «Технологии разработки качественного программного обеспечения»

Выполнил:

студент гр. В3530904/80322 Мирошниченко В. А.

Руководитель: Смирнов Н. Г.

Санкт-Петербург

2021

Оглавление

[1. Постановка задачи 3](#_Toc67250331)

[2. Описание архитектуры приложения 3](#_Toc67250332)

[3. Скриншоты приложения 5](#_Toc67250333)

[4. Оценка покрытия кода модульными тестами 6](#_Toc67250334)

[5. Перечисление использованных API и билиотек 6](#_Toc67250335)

[6. Выводы 6](#_Toc67250336)

# **Постановка задачи**

Необходимо разработать приложение «Словарь» на языке программирования Java. Данные словаря должны храниться в базе данных PostgreSQL. Приложение должно подключаться к базе данных через библиотеку JDBC и осуществлять поиск по нестрогому соответствию.

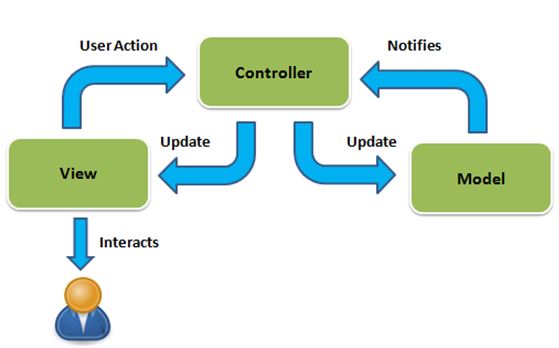
Так же приложение должно иметь возможность редактирования, добавления, удаления записей из базы данных и сохранять пользовательские настройки подключения, и иметь возможность редактирования файла, чтобы пользователю не вводить их каждый раз при запуске приложения.

# **Описание архитектуры приложения**

* 1. **Описание используемых паттернов проектирования**

Для реализации лабораторной работы были выбраны два паттерна проектирования:

MVC (Model-View-Controller) – основной паттерн проектирования используемый в реализации приложения.



**Рис. 1** – Схема принципа работы MVC

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем), просто предоставляя доступ к данным и управлению ими.

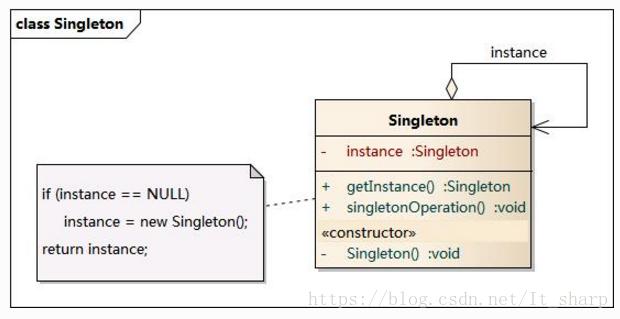
Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей».

Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

Для реализации взаимодействия программы с базой данных PostgreSQL разработан специальный класс «Database Manager», который был разработан с использованием паттерна проектирования Singleton.



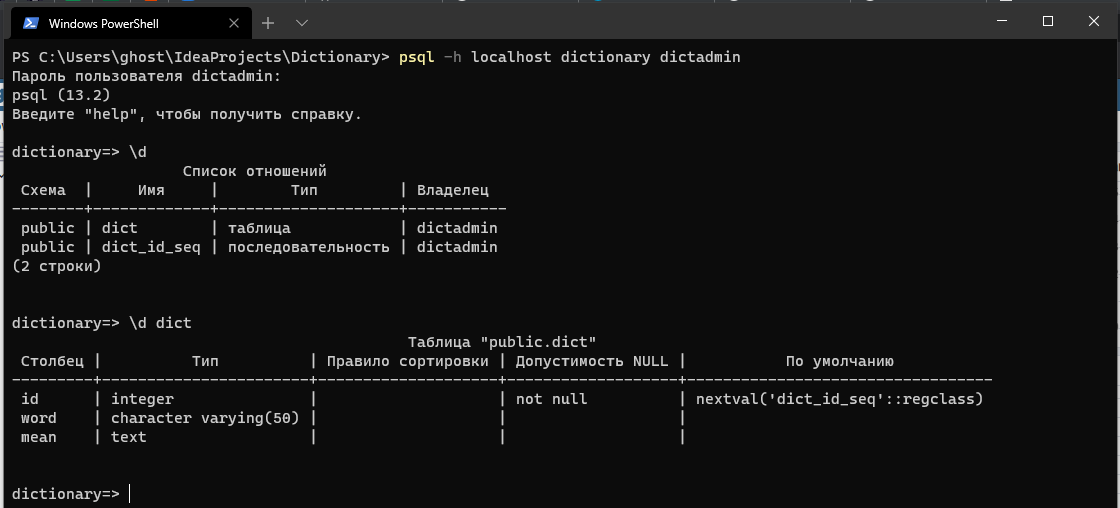
**Рис. 2** – Диаграмма класса паттерна Singleton

Singleton - порождающий паттерн, который гарантирует, что для определенного класса будет создан только один объект, а также предоставит к этому объекту точку доступа.

У класса есть только один экземпляр, и он предоставляет к нему глобальную точку доступа. При попытке создания данного объекта он создаётся только в том случае, если ещё не существует, в противном случае возвращается ссылка на существующий экземпляр и нового выделения памяти не происходит.

Существенно то, что можно пользоваться именно экземпляром класса, так как при этом во многих случаях становится доступной более широкая функциональность. Например, к описанным компонентам класса можно обращаться через интерфейс, если такая возможность поддерживается языком.

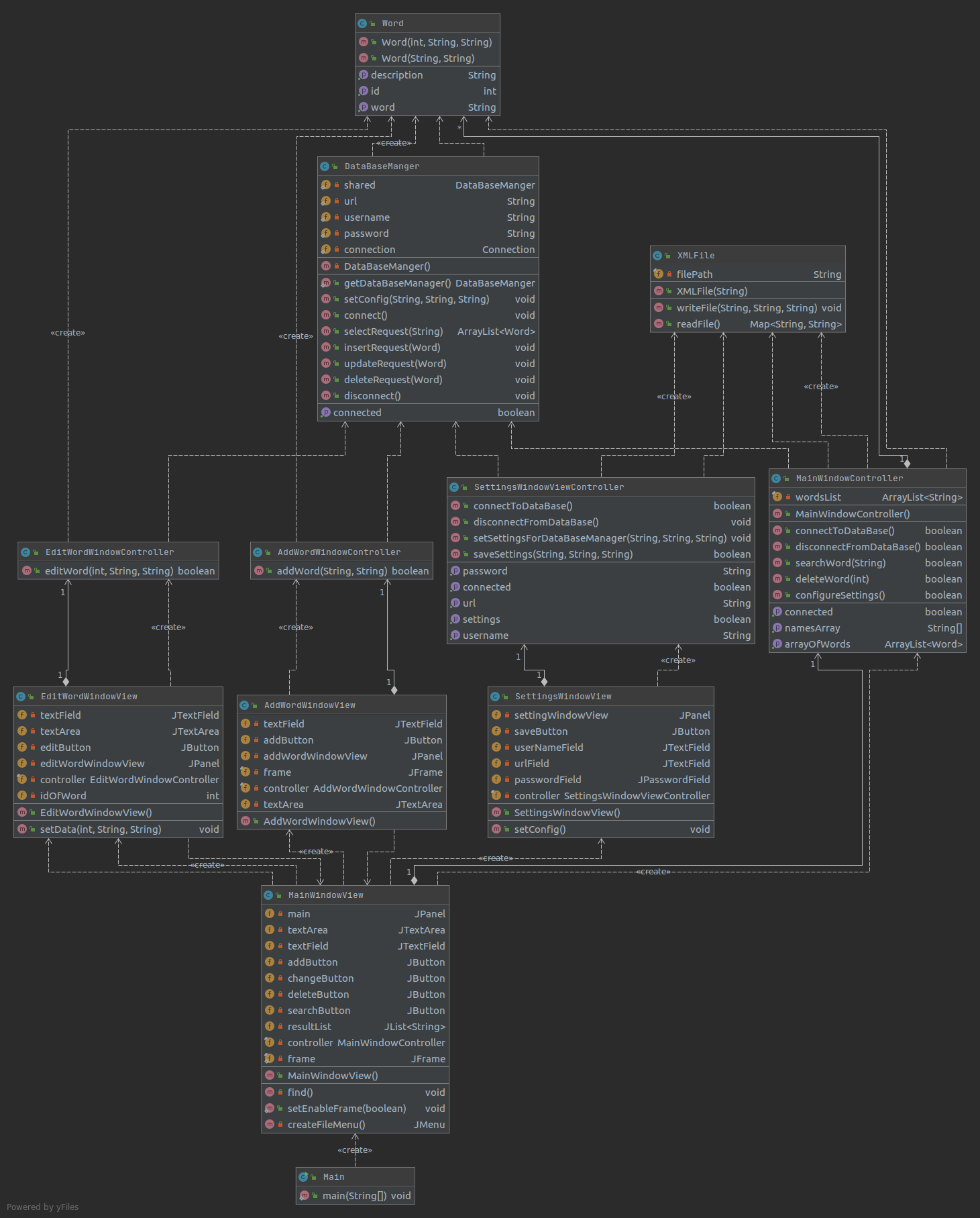
* 1. **Описание базы данных**



**Рис. 3** – Структура базы данных и таблицы

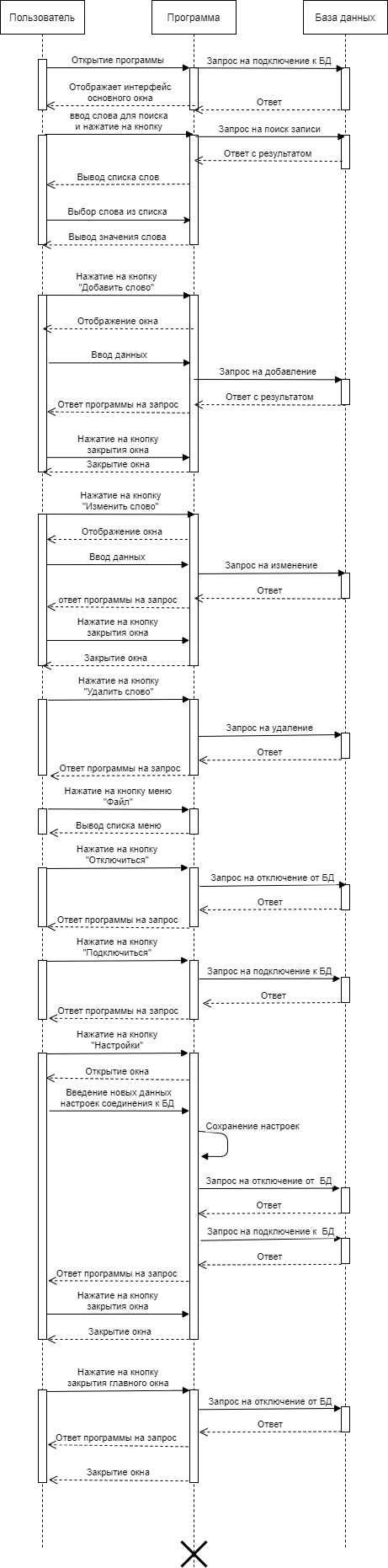
Таблица в базе данных PostgreSQL состоит из трех столбцов:

1. Id – столбец с целочисленным номером записи в таблице. Заполняется автоинкрементным способом при добавлении новой записи в таблицу.
2. Word – столбец предназначенный для хранения слова.
3. Mean – столбец предназначенный для хранения значения слова.
   1. **UML диаграмма классов**



**Рис. 4** – UML диаграмма классов

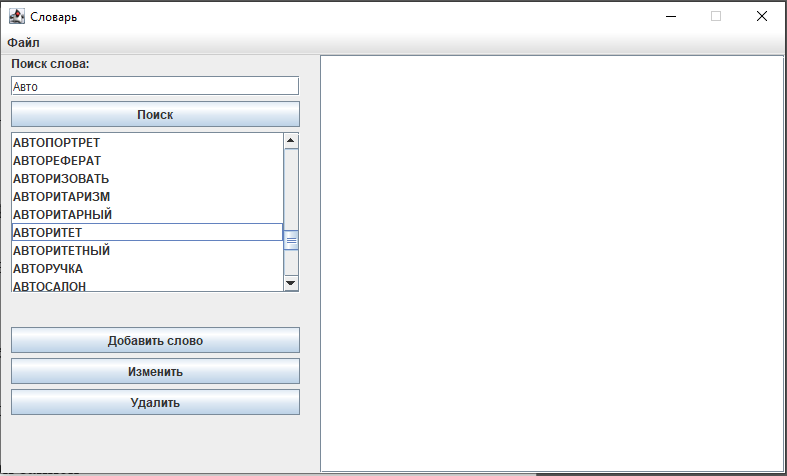
* 1. **Диаграмма последовательности**

****

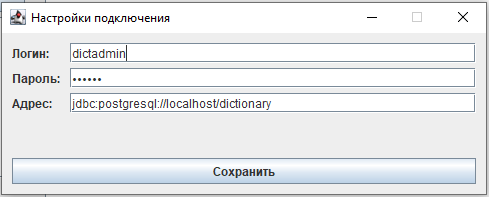
**Рис. 5** – Диаграмма последовательности

# **Скриншоты приложения**

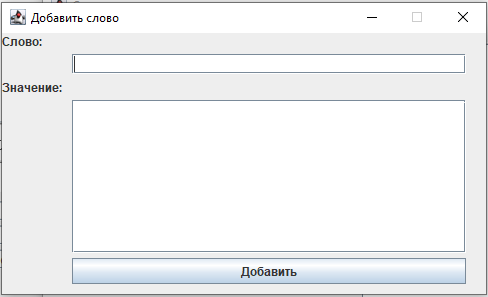
Скриншоты главных окон приложения:



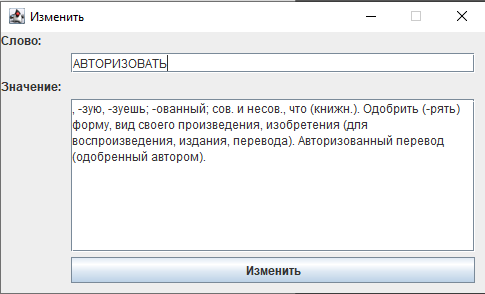
**Рис. 6** – Главное окно приложения



**Рис. 7** – Окно настроек подключения к базе данных



**Рис. 8** – Окно добавления новой записи



**Рис. 9** – Окно изменения выбранной записи

## **Перечисление использованных API и библиотек**

Для выполнения лабораторной работы использовались следующие API:

* JDBC - платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE.
* DOM (Document Object Model) - независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов. Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями «родительский-дочерний».

## **Выводы**

В ходе выполнения лабораторной работы было разработано приложение с пользовательским интерфейсом и взаимодействием с базой данных PostgreSQL.

Изучены способы взаимодействия Java приложения с базой данных PostgreSQL при помощи JDBC API. Изучены методы создания и редактирования XML файлов при помощи DOM API. Улучшены навыки в реализации и использовании паттернов MVC и Singleton. Изучены методы разработки пользовательского интерфейса для Java приложений с использованием фреймворка Swing.