Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине   
**«Интеграция кроссплатформенных ПС»**

*«Прототип программной системы анализа и   
визуализации данных педагогических работников»*

Выполнила:

студентка гр. ИСТм-120

Зеленцова А.А.

Принял:

Тимофеев А.А.

Владимир, 2022

**АННОТАЦИЯ**

В данной работе представлен прототип программной системы анализа и визуализации данных педагогических работников. Реализация прототипа включает в себя сервис взаимодействия с сокетами сервера и клиента, который получает данные из файла эксель и отправляет их на сервер, который в свою очередь предоставляет результат анализа данных в виде графиков.

Курсовой проект представлен на 22 страницах, рисунков – 15, использованных источников – 7, приложений – 1.

**ABSTRACT**

This paper presents a prototype of a software system for analyzing and visualizing data of teaching staff. The prototype implementation includes a service for interacting with server and client sockets, which receives data from the excel file and sends it to the server, which in turn provides the result of data analysis in the form of graphs.

The course project is presented on 22 pages, figures - 15, used sources - 7, appendices - 1.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc98110047)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc98110048)

[1.1 Цели и задачи, стоящие перед создаваемой системой 4](#_Toc98110049)

[1.2 Описание предметной области 4](#_Toc98110050)

[2 Диаграмма прецедентов 5](#_Toc98110051)

[2.1 Анализ бизнес-процесса 5](#_Toc98110052)

[2.2 Функциональные требования 7](#_Toc98110053)

[2.3 Нефункциональные требования 7](#_Toc98110054)

[3. РАЗРАБОТКА ПРОТОКОЛА ВЗАИМОДЕЙСТВИЯ 8](#_Toc98110055)

[4. ПРОЕКТИРОВАНИЕ СИСТЕМЫ 10](#_Toc98110056)

[Общая организация системы 10](#_Toc98110057)

[Проектирование модели данных 10](#_Toc98110058)

[5. РЕАЛИЗАЦИЯ СИСТЕМЫ 15](#_Toc98110088)

[5.1. Реализация функциональных компонентов системы 15](#_Toc98110089)

[5.2. Тестирование 15](#_Toc98110090)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc98110091)

[ПРИЛОЖЕНИЕ А ЛИСТИНГ ПРОГРАММНОЙ СИСТЕМЫ 21](#_Toc98110092)

# ВВЕДЕНИЕ

В современном мире нас окружает огромное количество информации, с чем связано повышение интереса к таким вопросам, как анализ и визуализация данных. Такой резкий скачок к потреблению визуальной информации связан с тем, что сознание не способно успеть воспринять, отсортировать и проанализировать весь поток информации. Однако без данных не будет никакого анализа и визуального представления. Поэтому необходимо очень требовательно и серьезно относиться к этому ресурсу. В интернет ресурсах можно найти довольно много информации по анализу педагогической деятельности, однако в системе, которой пользуется Владимирская область (Барс) такого модуля не предусмотрено.

Отсюда возникает необходимость в разработке системы, способной собирать данные полученные из системе БАРС (выгрузка происходит в формате эксель), анализировать их и предоставлять результат анализа в виде, понятном пользователю.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Цели и задачи, стоящие перед создаваемой системой

Целью, которая стоит перед создаваемой системой, является упрощение и сокращение временных затрат на анализ педагогической деятельности, который обычно связан с большим количеством бумажной волокиты и представлением в виде графиков. Для достижения поставленной цели необходимо решить следующие задачи:

а) Изучить особенности предметной области, которая относится к теме данного курсового проекта;

б) Провести проектирование интеграционного модуля;

в) Разработать прототип интеграционного модуля;

г) Разработать клиентскую часть систем;

д) Проанализировать полученные результаты работы программной системы.

## Описание предметной области

Модуль анализа педагогически деятельности» - это модуль для организации взаимодействия между двумя не связанными между собой приложениями. Например, каждый год администрация собирает результаты ЕГЭ и заносит их а автоматизированную систему БАРС. Просто смотря на все данные довольно тяжело проанализировать успехи учителя, в этом им поможет созданный модуль. Результаты ЕГЭ выкачиваются из БАРСа и анализируются, выдавая результат пользователю в виде графика, на котором виден прогресс определенного учителя.

## Диаграмма прецедентов

В системе существует два типа пользователей: администратор и сотрудник предприятия. Каждый из этих типов по-разному использует систему.

Различные варианты использования системы, отношения между актерами и прецедентами, представлены на диаграмме прецедентов, которая изображена на рисунке

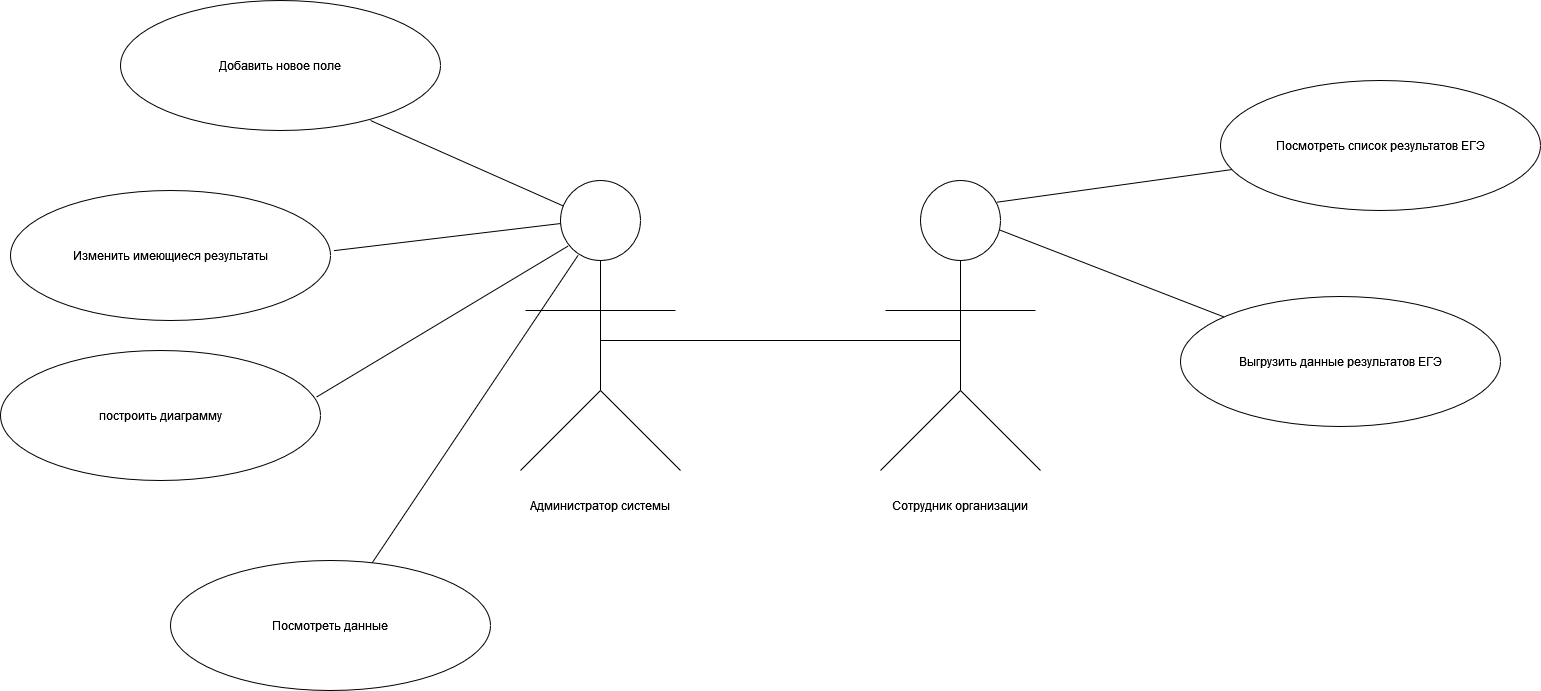


Рисунок 1 – Диаграмма прецедентов системы

* 1. Поиск (для обычных пользователей и администратора)

Параметры поиска:

* Рейтинг (оценка на основе данных ЕГЭ, стажа, данных опросов, проводимых директором среди учеников)
* По ФИО
  1. Загрузка данных (только для администратора)

Содержание:

* Таблица эксель с данными ЕГЭ
* Ручной ввод;
  1. Корректировка (только для администратора)

Параметры корректировки:

* Добавление новой информации, не имеющейся в системе
  1. Основные функциональные требования

Для организации

* Загрузка обновленных результатов ЕГЭ;
* Добавить/изменить информацию об учителе;
* Проанализировать деятельность учителя;
  1. Варианты использования
     1. Регистрация
* Действия: *незарегистрированный пользователь* не видит никакую информацию в системе;
* Для регистрации нужно пройти все действия указанные в руководстве по «БАРС».
  + 1. Вход
* Для входа нужно пройти все действия указанные в руководстве по «БАРС».
  + 1. Анализ деятельности учителя:

Администратор

* может нажать на определенного учителя чтобы посмотреть информацию об учителе, какие ученики у него учились, в каких годах
* может нажать на конкретного ученика, увидеть его результаты ЕГЭ
* увидеть аналитическую схему

Обычный пользователь (ученик, родитель)

* может увидеть только аналитическую схему

## Анализ бизнес-процесса

Основными процессами взаимодействия систем являются запрашивание данных через систему и возвращение проанализированного результата в виде графиков. BPMN – диаграмма представлена на рисунке 2.

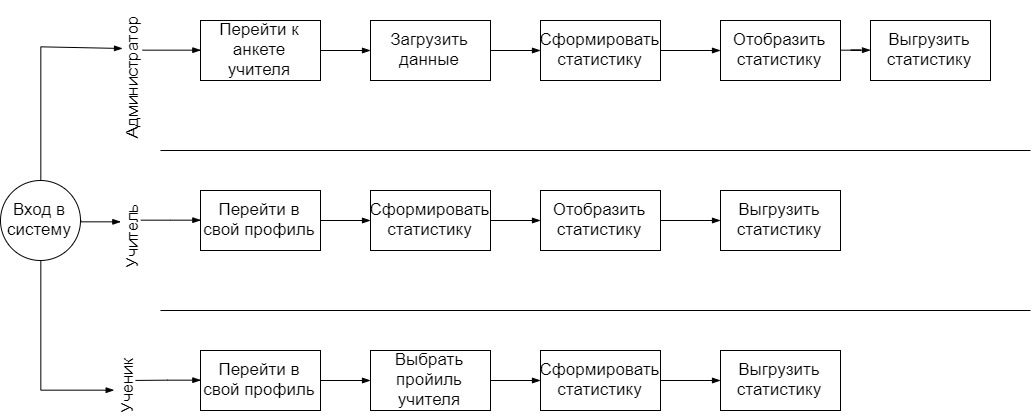


Рисунок 2 - BPMN – диаграмма

## Функциональные требования

Ниже описаны требования к функциональности продукта (Таблица 1).

Таблица 1 – Функциональные требования

|  |  |
| --- | --- |
| **Идентификатор** | **Описание** |
| 1 | Получение списка учителей |
| 2 | Получение списка учеников |
| 3 | Анализ данных |
| 4 | Предоставление результата анализа данных |

## Нефункциональные требования

Нефункциональные требования к системе представлены в таблице ниже.

Таблица 2 - Требования к программной совместимости

|  |  |
| --- | --- |
| **Идентификатор** | **Описание** |
| 1 | Среда разработки IntelliJ IDEA |
| 2 | Язык Java |

# 

# РАЗРАБОТКА ПРОТОКОЛА ВЗАИМОДЕЙСТВИЯ

Для передачи данных между двумя модулями был использован Socket и ServerSocket.

Наиболее распространенный протокол HTTP (Hyper Text Transfert Protocol) определяет взаимодействие двух программ, клиента и сервера, которые могут быть запущены на разных и удаленных друг от друга машинах. Клиентом является приложение, которое пользуется каким-то сервисом, предоставляемым сервером, обычно размещенном на удаленном компьютере. Клиент должен подключиться к удаленному серверу, который постоянно находится в режиме ожидания соединения. После этого они могут обмениваться информацией.

HTTP использует протокол TCP/IP. В статье рассмотриваются возможности, предоставляемые Java для работы с этим протоколом.

**Распределение протоколов по уровням модели TCP/IP**

|  |  |  |
| --- | --- | --- |
| **№ п.п.** | **Название** | **Протоколы** |
| 5 | Прикладной | HTTP, RTP, FTP, DNS |
| 4 | Транспортный | TCP, UDP, SCTP, DCCP |
| 3 | Сетевой | IP. Вспомогательные протоколы, вроде ICMP и IGMP, работают поверх IP. |
| 4 | Канальный | Ethernet, IEEE 802.11 Wireless Ethernet, SLIP, Token Ring, ATM и MPLS |
| 5 | Физический | Физическая среда и принципы кодирования информации, T1, E1 |

Протокол HTTP располагается на прикладном уровне и использует для собственной реализации протоколы более низких уровней. Основой HTTP является протокол транспортного уровня TCP.

Согласно протоколу IP (Internet Packet), каждый узел (компьютер, switch и т.п.) в сети имеет свой IP-адрес. На данный момент интернет работает по протоколу IPv4, где IP адрес записывается 4 числами от 0 до 255 - например, 127.0.0.1. Существует и другой способ идентификации компьютеров в сети через доменное имя, которое более удобное и нагляднее идентифицирует компьютер, чем простой набор чисел (например, java-oline.ru). В Интернете существуют специальные сервера DNS (Domain Name System), которые осуществляют преобразование доменного имени в IP-адрес и наоборот.

TCP протокол базируется на IP для доставки пакетов, но добавляет два важных свойства :

* установление соединения между приложениями;
* использование портов, а не просто узлов.

Таким образом, для идентификации компьютера (host'a) в сети используется IP-адрес; для идентификации приложения TCP добавляет понятие порта. Порт - это целое число от 1 до 65535 указывающее, какому приложению предназначается пакет.

Java для работы в сети имеет специальный пакет *java.net*, содержащий класс **Socket**, что в переводе означает «гнездо». Ключевыми классами для реализации взаимодействия программ по протоколу TCP являются :

* java.net.ServerSocket - класс реализует серверный сокет, который ожидает запросы, приходящие от клиентов по сети, и может отправлять ответ.
* java.net.Socket - класс реализует клиентский сокет.

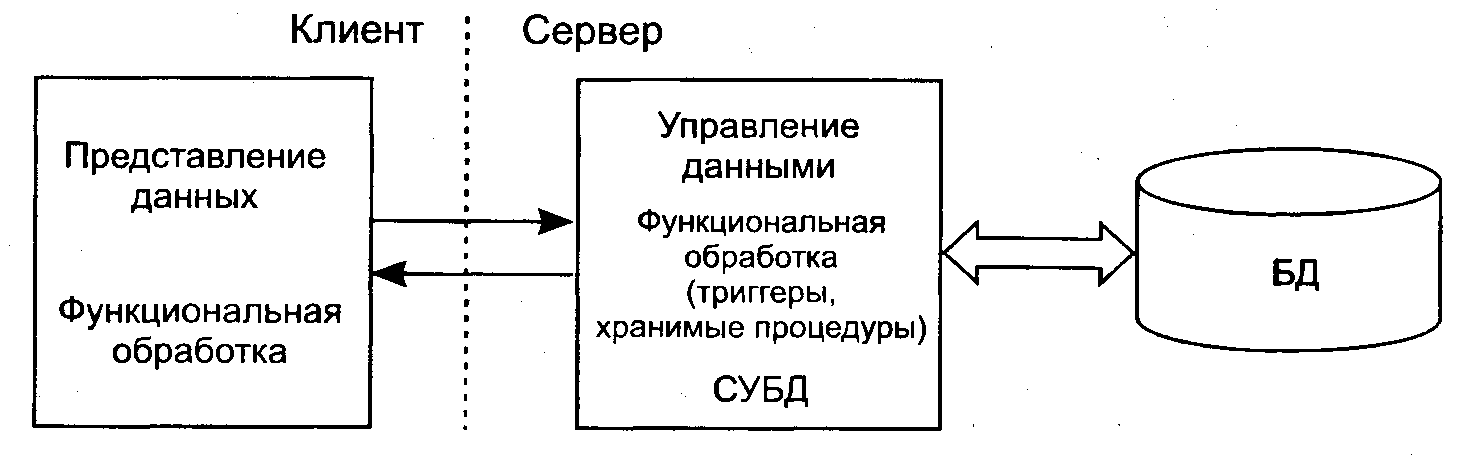


Рисунок 2 – Схема взаимодействия

# 

# ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## Общая организация системы

Разрабатываемая система будет состоять из двух основных компонентов. Система предприятия берет данные о результат ЕГЭ из БД и по требованию передает их прототипу модуля анализа, а также по требованию получает данные от этого прототипа модуля.

BPMN-диаграмма, обозначающая бизнес-процесс получения модулем списка сотрудников показана на рисунке 3.

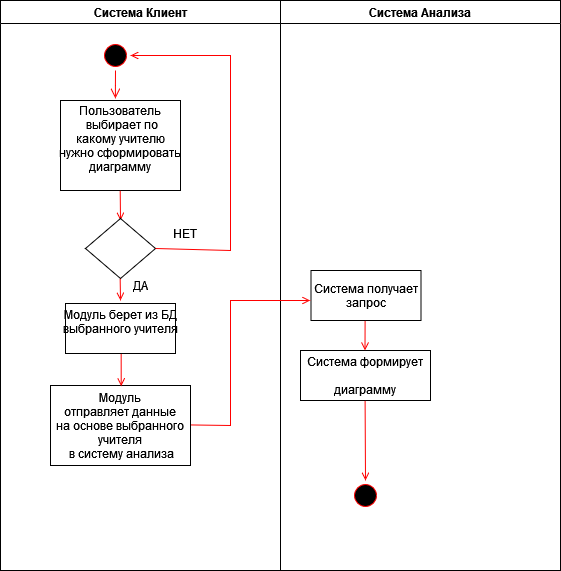


Рисунок 3 – BPMN-диаграмма формирования модулем диаграммы

## Проектирование модели данных

Всего во взаимодействии модулей участвует 4 таблицы, две относятся к модулю проведения экзамена, две к системе отдела кадров.

## Объявляется этот класс на стороне клиента, а сервер воссоздаёт его, получая сигнал на подключение. Так происходит общение в сети. Для начала вот возможные конструкторы класса Socket:

## Socket(String имя\_хоста, int порт) throws UnknownHostException, IOException

## Socket(InetAddress IP-адрес, int порт) throws UnknownHostException

## «имя\_хоста» — подразумевает под собой определённый узел сети, ip-адрес. Если класс сокета не смог преобразовать его в реальный, существующий, адрес, то сгенерируется исключение UnknownHostException.

## Порт — есть порт. Если в качестве номера порта будет указан 0, то система сама выделит свободный порт. Также при потере соединения может произойти исключение IOException.

## Следует отметить тип адреса во втором конструкторе — InetAddress. Он приходит на помощь, например, когда нужно указать в качестве адреса доменное имя. Так же когда под доменом подразумевается несколько ip-адресов, то с помощью InetAddress можно получить их массив. Тем не менее с ip он работает тоже. Так же можно получить имя хоста, массив байт составляющих ip адрес и т.д. Мы немного затронем его далее, но за полными сведениями придется пройти к официальной документации.

## При инициализации объекта типа Socket, клиент, которому тот принадлежит, объявляет в сети, что хочет соединиться с сервером про определённому адресу и номеру порта. Ниже представлены самые часто используемые методы класса Socket:

## InetAddress getInetAddress() – возвращает объект содержащий данные о сокете. В случае если сокет не подключен – null

## int getPort() – возвращает порт по которому происходит соединение с сервером

## int getLocalPort() – возвращает порт к которому привязан сокет. Дело в том, что «общаться» клиент и сервер могут по одному порту, а порты, к которым они привязаны – могут быть совершенно другие

## boolean isConnected() – возвращает true, если соединение установлено

## void connect(SocketAddress адрес) – указывает новое соединение

## boolean isClosed() – возвращает true, если сокет закрыт

## boolean isBound() - возвращает true, если сокет действительно привязан к адресу

## Класс Socket реализует интерфейс AutoCloseable, поэтому его можно использовать в конструкции try-with-resources. Тем не менее закрыть сокет также можно классическим образом, с помощью close().

## Голова Три: а это ServerSocket

## Допустим мы объявили, в виде класса Socket, на стороне клиента запрос на соединение. Как сервер разгадает наше желание? Для это сервер имеет такой класс как ServerSocket, и метод accept() в нём. Его конструкторы представлены ниже:

## ServerSocket() throws IOException

## ServerSocket(int порт) throws IOException

## ServerSocket(int порт, int максимум\_подключений) throws IOException

## ServerSocket(int порт, int максимум\_подключений, InetAddress локальный\_адрес) throws IOException

## При объявлении ServerSocket не нужно указывать адрес соединения, потому что общение происходит на машине сервера. Только при многоканальном хосте нужно указать к какому ip привязан сокет сервера.

## Так как предоставлять программе больше ресурсов чем ей необходимо - и затратное и не разумное дело, поэтому в конструкторе ServerSocket вам предлагают объявить максимум соединений, принимаемых сервером при работе. Если оно не указано, то умолчанию это число будет считаться равным 50.

## Да, по идее можно предположить, что ServerSocket это такой же сокет, только для сервера. Но он играет совершенно иную роль нежели класс Socket. Он нужен только на этапе создания соединения.

## Создав объект типа ServerSocket необходимо выяснить, что с сервером кто-то хочет соединиться. Тут подключается метод accept(). Искомый ждёт пока кто-либо не захочет подсоединится к нему, и когда это происходит возвращает объект типа Socket, то есть воссозданный клиентский сокет. И вот когда сокет клиента создан на стороне сервера, можно начинать двухстороннее общение.

## Создать объект типа Socket на стороне клиента и воссоздать его с помощью ServerSocket на стороне сервера – вот необходимый минимум для соединения.Анкета ученика. Все данные ученик может только просмотреть.

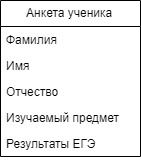


Рисунок 4 – Анкета ученика

## Анкета учителя. Вся анкета заполняется администратором. У учителя есть возможность отредактировать поля ФИО. Остальные данные учитель может только просмотреть.

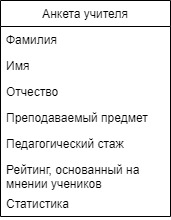


Рисунок 5 – Анкета учителя

## Администратор. Имеет возможность редактирования любых полей.

## Администратор

Рисунок 6 – Возможности администратора

1. РЕАЛИЗАЦИЯ СИСТЕМЫ

## Реализация функциональных компонентов системы

Для сервиса определена следующая структура:

* SistemAnalize – запрашивает данные у клиента.
* ReaderExcelDemo – считывает данные из БД.
* Сlient – вводит данные и отправляет SistemAnalize.
* UpdateExcelDemo – новые строки в эксель.
* JavaToMySQL – чтение данных из MuSQL и запись в эксель.
* JavaToMySQLpast – добавление новых строк в MySQL.

Структура сервиса представлена на рисунке 5.

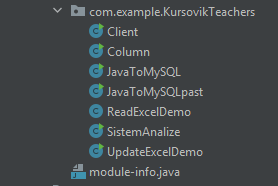


Рисунок 7 - Структура

## Тестирование

Для тестирования сервиса были использованы возможности HTTP-клиента в IntelliJ IDEA, позволяющие создавать, редактировать и выполнять HTTP-запросы. Было произведено тестирование всего функционала сервиса. Пример тестирования методов представлены ниже.

После того как мы записали данные в БД, мы можем их считать

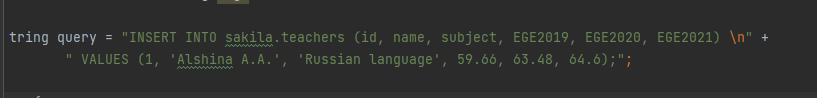


Рисунок 8 – данные для записи в MySQL

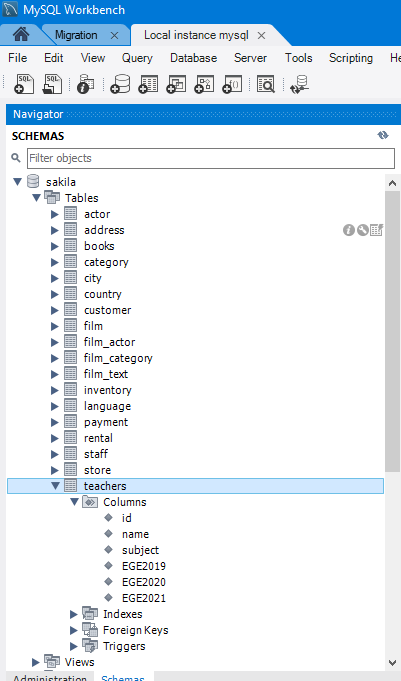


Рисунок 9 – данные в MySQL

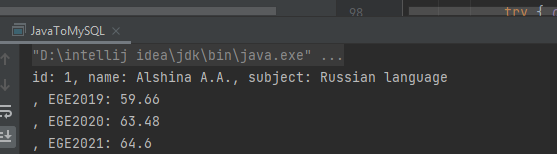


Рисунок 10 – данные из MySQL и запись в эксель

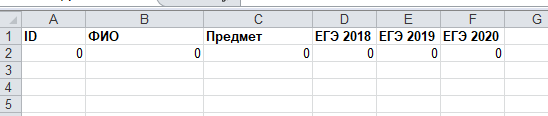


Рисунок 11 – эксель до записи

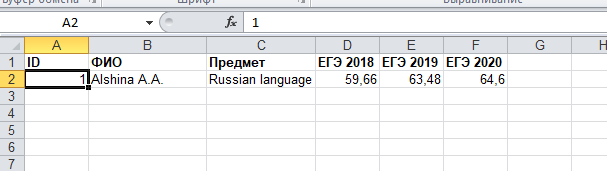


Рисунок 12 – эксель после записи

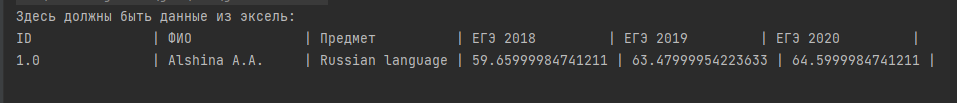


Рисунок 13 – данные со стороны клиента

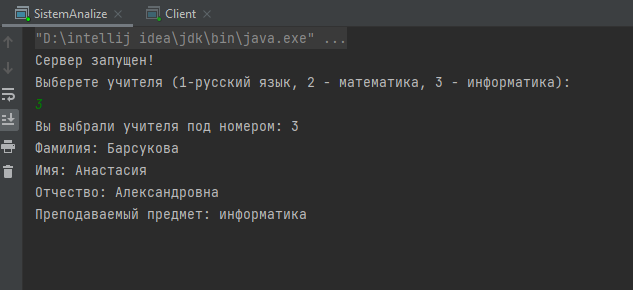


Рисунок 14 – выбор учителя

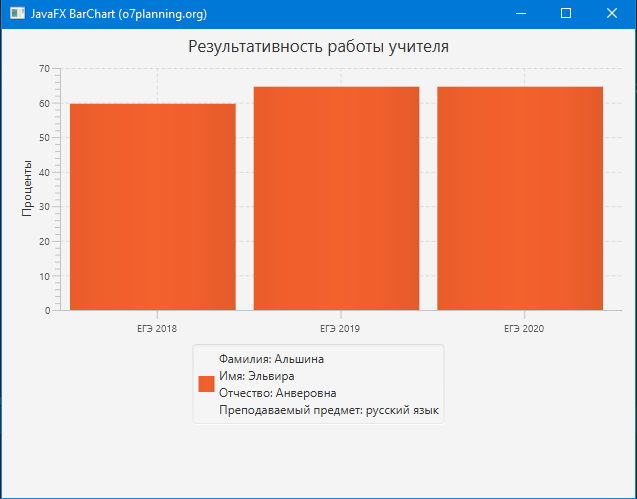


Рисунок 15 – анализ результатов в виде диаграммы

ЗАКЛЮЧЕНИЕ

В ходе работы были реализованы интеграционный модуль и прототип модуля проведения экзамена. Разработанный модуль выполняет свою основную функцию, а именно передачу данных прототипу модуля проведения экзамена и получение данных от него же.

Но интеграционный модуль еще может улучшаться и модифицироваться для расширения функционала.

Были выполнены все задачи, а именно:

* Анализ предметной области;
* Разработка протокола взаимодействия;
* Проектирование системы;
* Реализация системы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. JavaRush — URL: https://javarush.ru/groups/posts/654-klassih-socket-i-serversocket-ili-allo-server-tih-menja-slihshishjh
2. betacode.net — URL: [https://betacode.net/11091/javafx-button#a3728623](https://betacode.net/11091/javafx-button#a3728623 )
3. fandom — URL: <https://jfx.fandom.com/ru/wiki/JavaFX_Community>
4. Java-online.ru — URL: <https://java-online.ru/java-socket.xhtml>
5. metanit.com. Сайт о программировании. — URL: https://metanit.com/java/javafx/1.4.php
6. ГОСТ 2.105-95. Общие требования к текстовым документам. Единая система конструкторской документации. Общие требования к текстовым документам. Минск: Межгосударственный совет по стандартизации, метрологии и сертификации, 2006. – 95с.
7. ГОСТ 7.82 – 2001. Библиографическая запись. Библиографическое описание электронных ресурсов. – Минск: Межгосударственный совет по стандартизации, метрологии и сертификации, 2001. – 23с.

ПРИЛОЖЕНИЕ А  
ЛИСТИНГ ПРОГРАММНОЙ СИСТЕМЫ

Полный код проекта доступен по ссылке:

https://github.com/Arina7628/KursovikTeachers