Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.03.04 Программная инженерия

Профиль Проектирование и разработка программного обеспечения

**КУРСОВОЙ ПРОЕКТ**

по дисциплине             Проектироание информионных систем

(наименование дисциплины)

на тему \_Информационная система «Абитуриент»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(тема курсового проекта )

*Выполнил студент* ЗФО *курса*\_\_3\_\_\_\_*группы*\_\_20-ЗКБс-ПР1\_\_

       \_\_\_Шарангия Игорь Сергеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ф.и.о.)

Допущен к защите\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель (нормоконтролер) работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Янаева М.В.

Защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата, расшифровка подписи)

Краснодар

2023

ФГБОУ ВПО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт компьютерных систем и информационной безопасности\_\_\_\_\_\_\_

Кафедра информационных систем и программирования \_\_\_\_\_\_\_\_\_

Направление подготовки \_\_\_09.03.04 Программная инженерия\_\_\_\_\_\_\_\_\_\_

Профиль Проектирование и разработка программного обеспечения \_\_\_\_

УТВЕРЖДАЮ

Зав. кафедрой \_\_\_\_\_\_М.В. Янаева.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**ЗАДАНИЕ**

на курсовой проект

Студенту:\_Шарангия Игорь Сергееич\_\_ группы\_\_\_20-ЗКБс-ПР1\_\_ 3\_ курса

(Ф.И.О.) (№ группы и курса)

Тема проекта:\_ Информационная система «Абитуриент»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(утверждена указанием директора института № \_\_ от \_\_\_\_ 20 \_\_\_\_)

План проекта:

1. \_Исследование предметной области\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. \_Спецификация требований\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.\_\_Проектирование информационной системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4.\_\_Реализация информационной системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Объем работы:

а) пояснительная записка \_\_\_\_\_\_\_\_\_\_\_\_\_ с.

б) листинг \_\_\_\_\_\_\_\_\_\_\_\_\_ лист (-ов)

Рекомендуемая литература

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Срок выполнения: с «\_\_\_» \_\_\_\_\_\_по «\_\_\_»\_\_\_\_20\_\_г.*

Срок защиты:                      «\_\_\_»\_\_\_\_20\_\_г.

Дата выдачи задания:                «\_\_\_»\_\_\_\_20\_\_г.

Дата сдачи проекта на кафедру:       «\_\_\_»\_\_\_\_20\_\_г.

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Янаева М.В.

(должность, подпись,)

Задание принял студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шарангия И.С.

(подпись)

**Реферат**

Курсовой проект: 38 страниц, 38 рисунка, 8 листинга, 7 используемых источников, 3 приложения.

ИНФОРМАИОННАЯ СИСТЕМА АБИТУРИЕНТ, БАЗЫ ДАННЫХ, УПРАВЛЕНИЕ ПРОЕКТАМИ, диаграмма классов, диаграмма взаимодействия ДАННЫХ, ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ.

Объектом является деятельность приемной комиссии учебного заведения для организации работы с документами абитуриента.

Целью работы является увеличение эффективности работы за счёт обеспечения на техническом уровне следующих условий: стандартизация процессов работы с абитуриентом; наличие единого хранилища информации; выдача отчетной документации по вступительным испытаниям абитуриентов.

Основные полученные результаты:

* проведён анализ ключевых процессов работы с абитуриентами, позволяющих автоматизацию в краткосрочной перспективе.
* спроектирована и разработана информационная система для учебного заведения;

Об эффективности внедрения можно судить по изменению таких показателей, как: время обработки обращения абитуриента, время на подготовку отчётов о вступительных испытаниях.

Содержание

[Введение 6](#_Toc130172136)

[1 Описание предметной области 7](#_Toc130172137)

[1.1 Общее описание предметной области 7](#_Toc130172138)

[1.2 Назначение и цели создания ПО 7](#_Toc130172139)

[1.3 Требования к структуре ПО и базе данных 7](#_Toc130172140)

[1.4 Описание входных документов и сообщений 8](#_Toc130172141)

[1.5 Описание выходных документов и сообщений 8](#_Toc130172142)

[1.6 Список ограничений 9](#_Toc130172143)

[2 Проектирование структуры базы данных 9](#_Toc130172144)

[2.1 Описание сущностей 9](#_Toc130172145)

[2.2 Построение концептуальной схемы базы данных 10](#_Toc130172146)

[3 Реализация базы данных средствами выбранной СУБД 10](#_Toc130172147)

[3.1Выбор СУБД 10](#_Toc130172148)

[3.2 Проектирование таблиц 11](#_Toc130172149)

[3.3 Построение ER- диаграммы 15](#_Toc130172150)

[4 Проектирование структуры приложения 16](#_Toc130172151)

[4.1 Построение структурной схемы ПО 16](#_Toc130172152)

[4.2 Модель функционирования системы 17](#_Toc130172153)

[4.3 Диаграмма компонентов 17](#_Toc130172154)

[4.4 Диаграммы кооперации 18](#_Toc130172155)

[4.5 Диаграмма размещения 19](#_Toc130172156)

[4.6 Диаграмма состояний 20](#_Toc130172157)

[4.7 Диаграммы потоков данных 20](#_Toc130172158)

[4.8 Диаграмма классов 21](#_Toc130172159)

[4.9 Диаграмма последовательности 22](#_Toc130172160)

[4.10 Диаграмма прецедентов 22](#_Toc130172161)

[5 Реализация приложения базы данных 23](#_Toc130172162)

[5.1 Выбор среды программирования 23](#_Toc130172163)

[5.2 Схема меню приложения 23](#_Toc130172164)

[5.3 Окна приложения или руководство пользователя 24](#_Toc130172165)

[5.4 Запросы к базе данных 29](#_Toc130172166)

[5.5 Функции и процедуры обработки данных 30](#_Toc130172167)

[5.6 Формирование отчетной документации 31](#_Toc130172168)

[6 Тестирование программного продукта 34](#_Toc130172169)

[Заключение 37](#_Toc130172170)

[Список использованных источников 38](#_Toc130172171)

[Приложение А Текст программы 39](#_Toc130172172)

[Приложение Б Текст модульных тестов 57](#_Toc130172173)

[Приложение В Антиплагиат 62](#_Toc130172174)

# Введение

В настоящее время образовательные учреждения стремятся не только обеспечить своим студентам высококачественное обучение, но и привлечь к себе множество новых абитуриентов. В связи с этим возникает необходимость в разработке информационной системы для упрощения процесса поступления.

Цель данного курсового проекта - разработка информационной системы абитуриент. Он описывает все этапы создания программного обеспечения, начиная от анализа требований и заканчивая реализацией и тестированием.

В работе проводится анализ предметной области, на основе которой определяются требования, которые необходимо реализовать в проекте. После этого формулируются функциональные и нефункциональные требования, которые должна выполнять система.

Дальше описываются этапы проектирования системы, включая разработку ее структурной схемы и алгоритмов. В конце проводится реализация и тестирование системы.

Данный проект имеет большую практическую значимость, поскольку он поможет упростить процесс поступления для абитуриентов и облегчит работу персонала приемной комиссии.

# 1 Описание предметной области

## 1.1 Общее описание предметной области

Предметная область курсового проекта "Абитуриент" связана с процессом поступления в вузы. Это включает в себя весь набор мероприятий и процедур, которые абитуриенты должны пройти, чтобы стать студентом высшего учебного заведения.

В рамках предметной области следует рассмотреть следующие аспекты:

- описание процесса подачи документов на поступление;

- информация о требованиях, которым должен соответствовать абитуриент при подаче документов;

- описание процедур вступительных испытаний: какие виды испытаний проходят абитуриенты.

- информация о заявлениях абитуриентов, необходимых для поступления в вуз;

## 1.2 Назначение и цели создания ПО

Целью курсового проекта является сбор и хранение информации об абитуриентах и результатах их вступительных испытаний для составления отчета с рейтингом на поступление.

## 1.3 Требования к структуре ПО и базе данных

Требования к структуре ПО важны для обеспечения эффективной разработки, тестирования, сопровождения и совместной работы над ПО. Хорошо спроектированная структура ПО может упростить разработку, повысить качество и надежность программного обеспечения и ускорить его время на рынке.

Для информационной системы «Абитуриент», функции, которые должна выполнять система, включают прием и хранение персональных данных абитуриента, создание заявления на поступление, хранение результатов вступительных испытаний и создание отчетов по вступительным испытаниям.

Требования к БД включают в себя сохранность данных, максимальную производительность и скорость доступа к данным, унификацию данных, возможность масштабирования БД для управления растущим потоком данных, использование открытых стандартов и простоту обращения с данными.

## 1.4 Описание входных документов и сообщений

Входными документами для системы являются следующие данные:

информация о специальностях, формах обучения, количестве мест, формах оплаты и дисциплинах;

персональные данные абитуриента, такие как ФИО, паспортные данные и СНИЛС;

заявления, которые подают абитуриенты на поступление;

результаты вступительных испытаний.

## 1.5 Описание выходных документов и сообщений

Описание выходных документов и сообщений будет содержать в себе все таблицы, а также:

Система будет генерировать отчеты по вступительным испытаниям абитуриентов.

Будет сформирован отчет с рейтингом абитуриентов по результатам вступительных испытаний.

## 1.6 Список ограничений

Администратор может создавать пользователей для сотрудников

Сотрудники могут вносить и изменять данные об абитуриентах, специальностях, заявлений, и результатов вступительных испытаний, создавать пользователей для абитуриентов.

Абитуриент может просматривать свои результаты вступительных испытаний и отчет с рейтингом.

# 2 Проектирование структуры базы данных

## 2.1 Описание сущностей

Для хранения данных в системе будут использоваться основные сущности: "Человек", "Вакансии" и "Заявления".

Сущность "Человек" будет содержать информацию об абитуриенте, такую как ФИО, паспортные данные и СНИЛС, который будет использоваться в качестве логина для входа в систему.

Сущность "Вакансии" будет хранить информацию о специальности, количестве доступных мест и дисциплинах вступительных испытаний.

Для хранения названий дисциплин будет использоваться отдельная сущность "Дисциплины".

Сущность "Заявления" будет содержать информацию о выбранной специальности и форме обучения абитуриента, а также может содержать несколько заявлений для одного абитуриента.

Результаты вступительных испытаний будут храниться в сущности "Экзамены".

Формы обучения и виды оплаты обучения будут храниться в отдельных сущностях "Форма обучения" и "Оплата обучения" соответственно.

Названия специальностей будут храниться в сущности "Специальность".

## 2.2 Построение концептуальной схемы базы данных

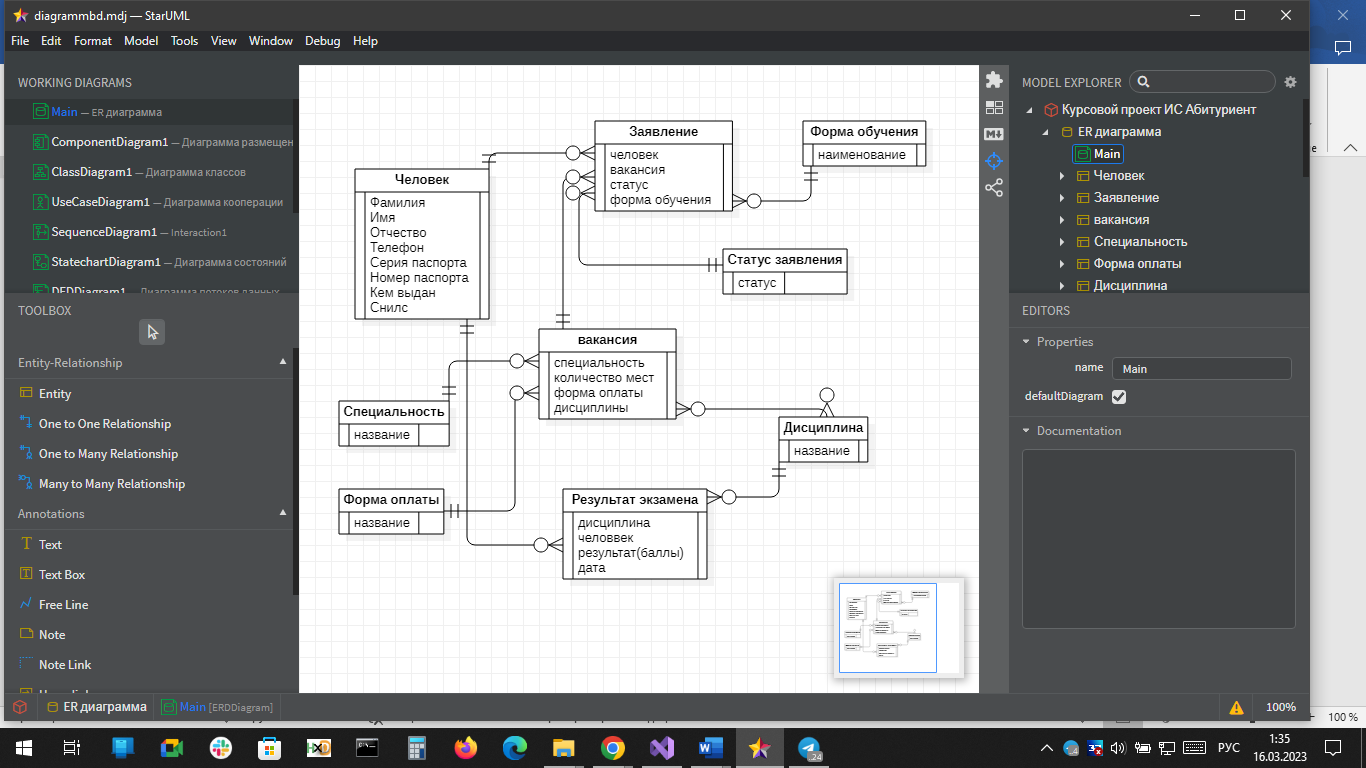


Рисунок 1 –Концептуальная схема бд

# 3 Реализация базы данных средствами выбранной СУБД

## 3.1Выбор СУБД

В качестве СУБД была выбрана PostgreSQL из-за ее открытости, надежности, высокой производительности и масштабируемости. PostgreSQL также имеет богатый набор функций, которые позволяют использовать его для широкого спектра приложений, включая веб-сайты, приложения для мобильных устройств, IoT и аналитические приложения. Кроме того, PostgreSQL имеет активную сообщество, которое работает над улучшением и поддержкой СУБД, что делает ее еще более привлекательной для разработчиков и предприятий.

## 3.2 Проектирование таблиц

Добавим таблицу для дисциплин, на рисунке 2.

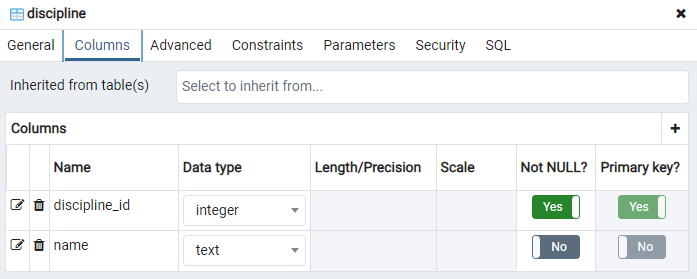


Рисунок 2 –Таблица дисциплин

Далее добавим таблицу специальности, на рисунке 3.

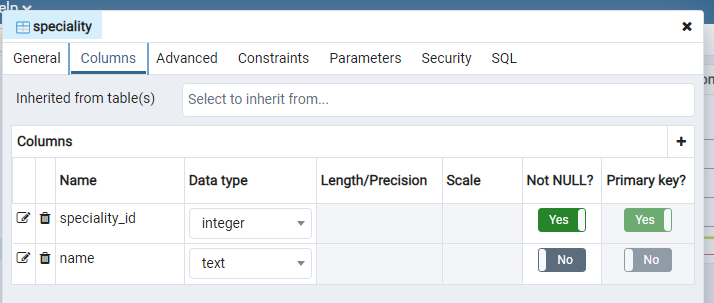


Рисунок 3 –Таблица специальности

Затем добавим таблицу для вакантных мест, рисунок 4.

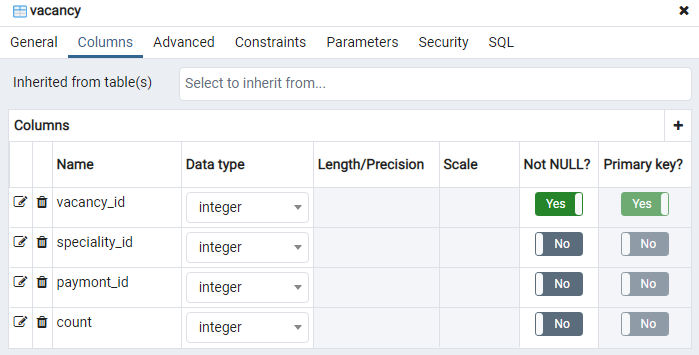


Рисунок 4 –Таблица вакантных мест

Далее добавим таблицу, с формами обучения, рисунок 5.

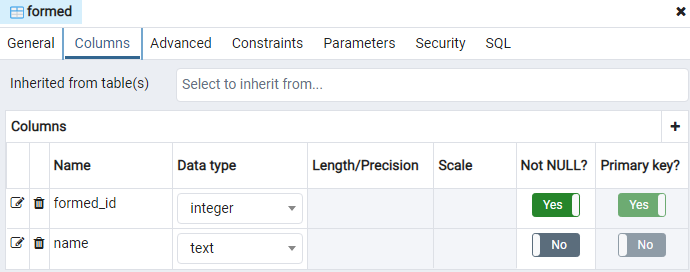


Рисунок 5 –Таблица формы обучения

Добавим таблицу с персональными данными, рисунок 6.

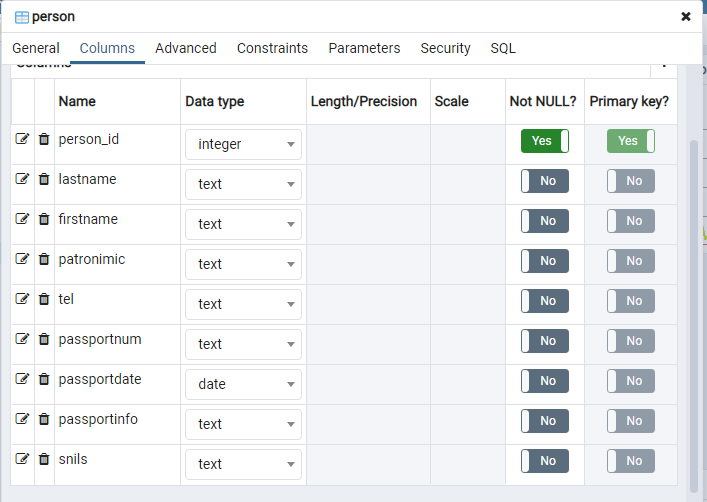


Рисунок 6 –Таблица персональных данных

Добавим таблицу с результатами экзаменов, рисунок 7.

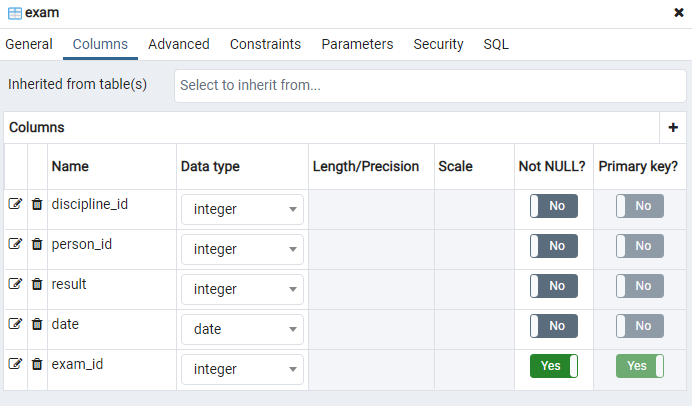


Рисунок 7 –Таблица результатов экзаменов

Добавим таблицу с заявлением абитуриента, рисунок 8.

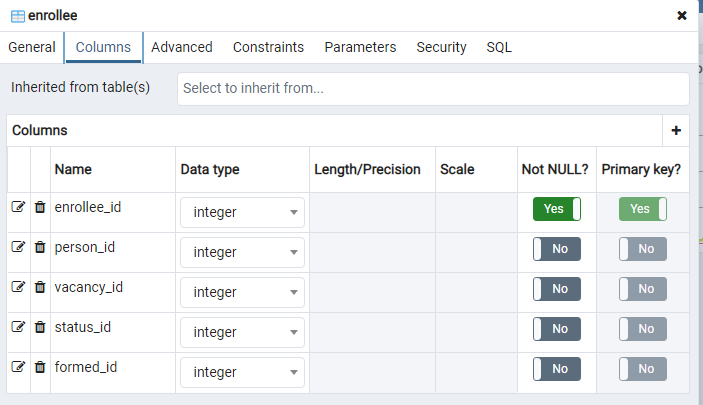


Рисунок 8 –Заявление абитуриента

Добавим таблицу дисциплин, рисунок 9.

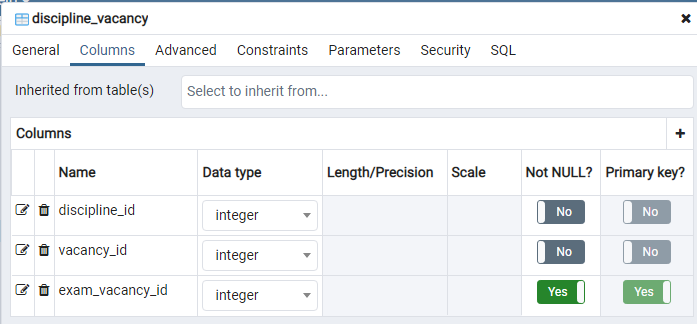


Рисунок 9 –Таблица необходимых дисциплин

Добавим таблицу с формой оплаты, рисунок 10.

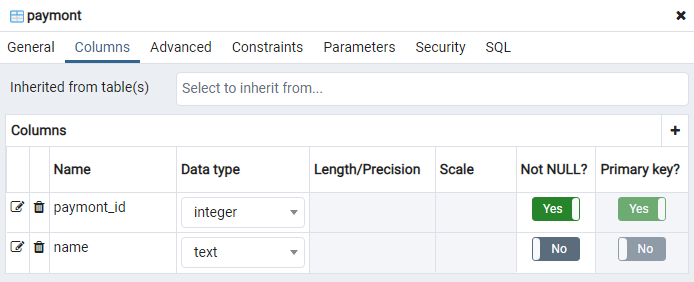


Рисунок 10 –Таблица оплата

Добавим таблицу статуса заявления, рисунок 11.

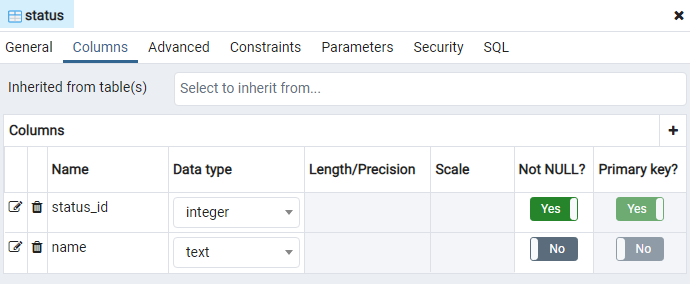


Рисунок 11–Таблица статус

## 3.3 Построение ER- диаграммы

Из созданных нами таблиц построим ER-диаграмму, создав необходимые связи, представлено на рисунке 12.

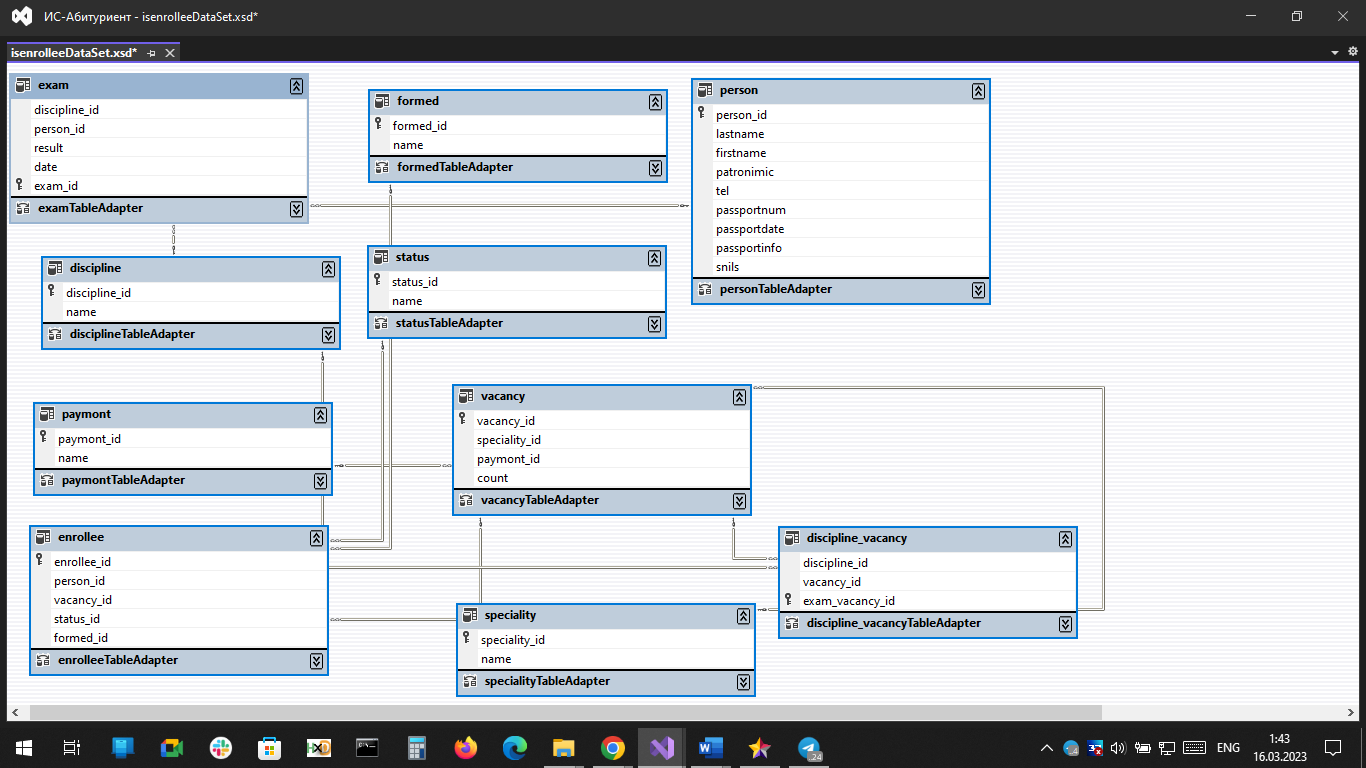


Рисунок 12 –ER-диаграмма

# 4 Проектирование структуры приложения

## 4.1 Построение структурной схемы ПО

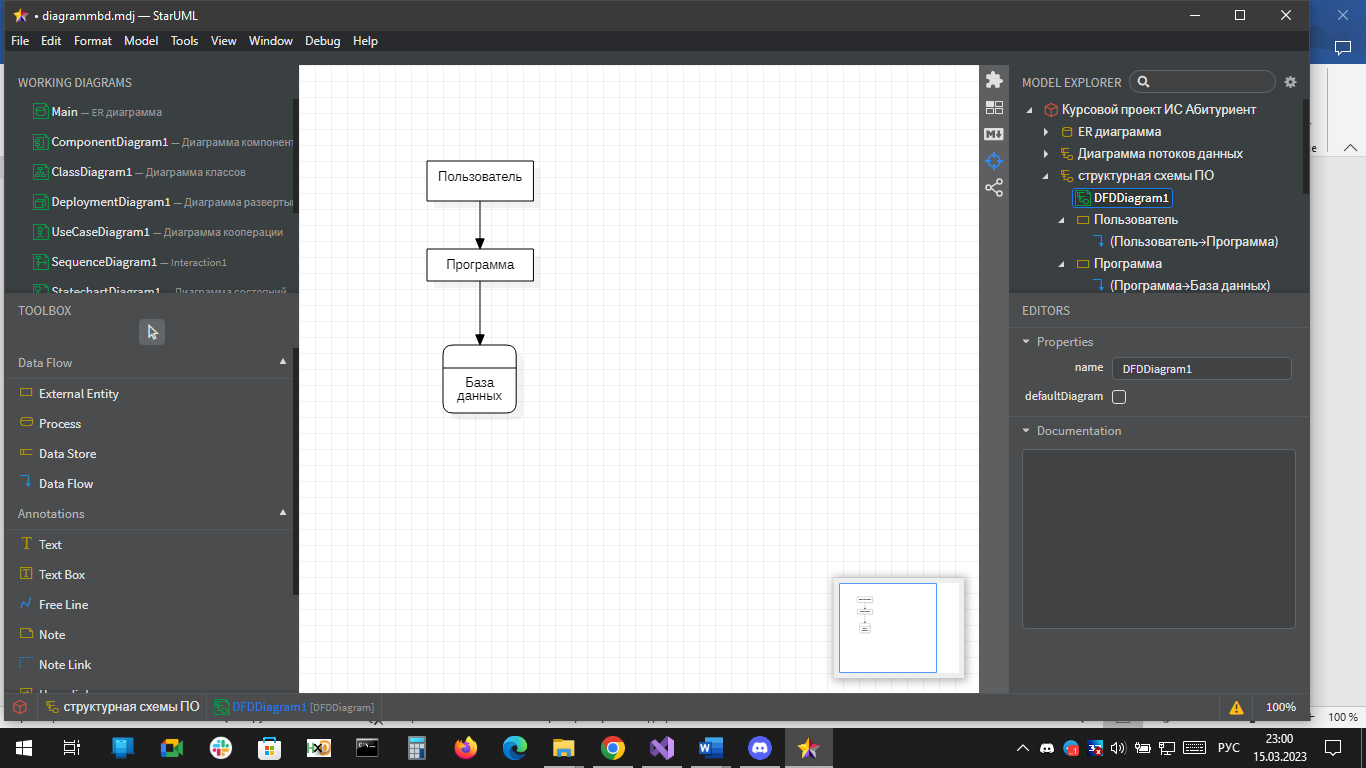


Рисунок 13 –Структурная схема

## 4.2 Модель функционирования системы

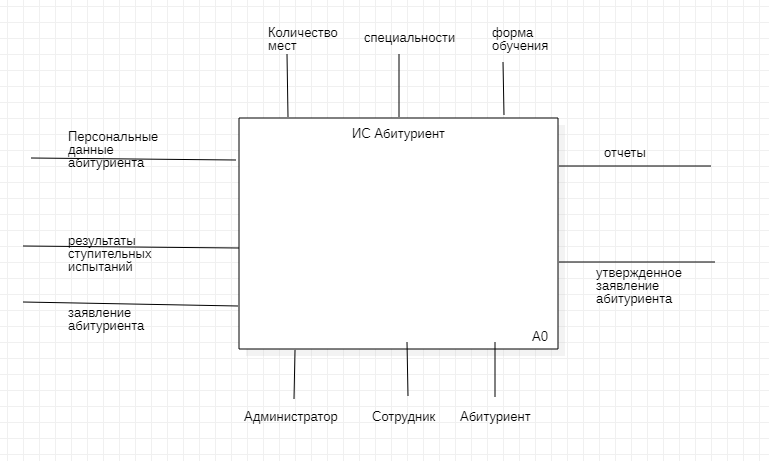


Рисунок 14 –Модель функционирования системы

## 4.3 Диаграмма компонентов

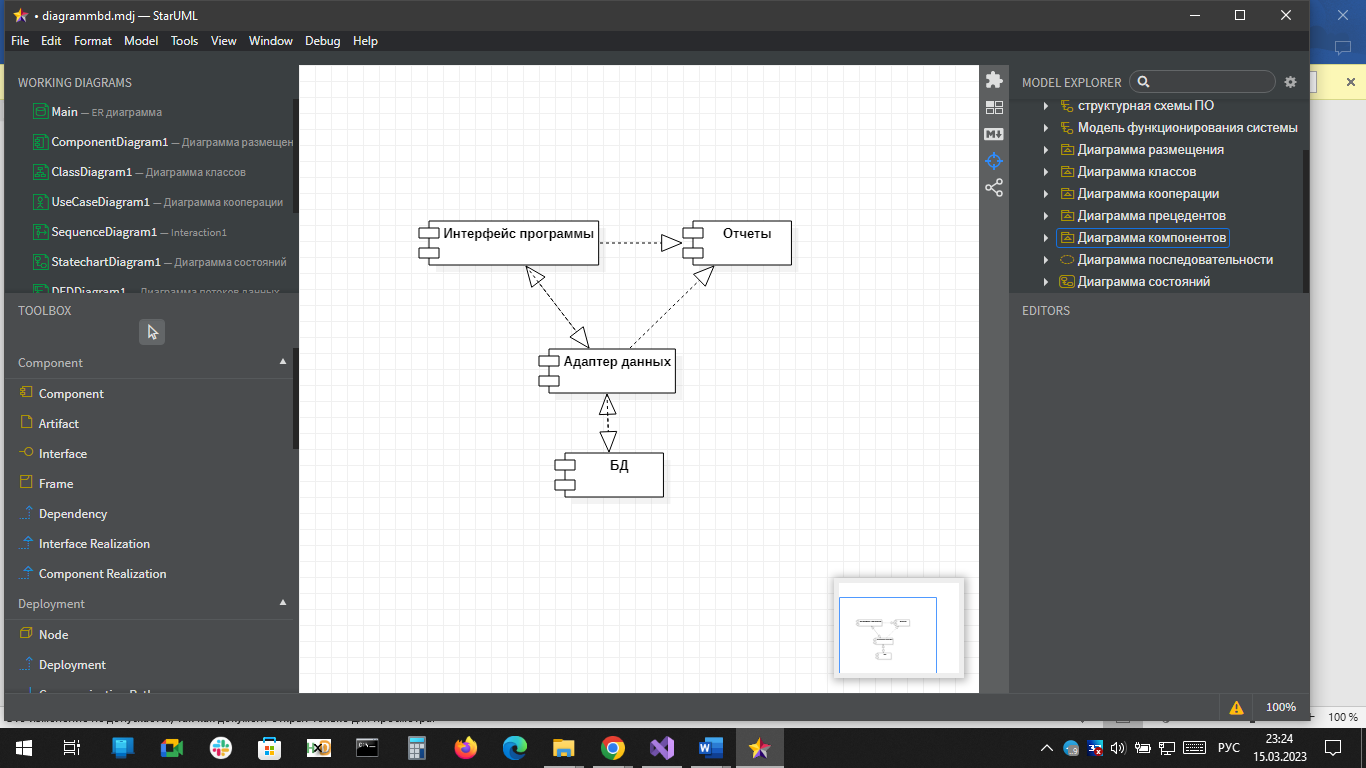


Рисунок 15 –Диаграмма компонентов

## 4.4 Диаграммы кооперации

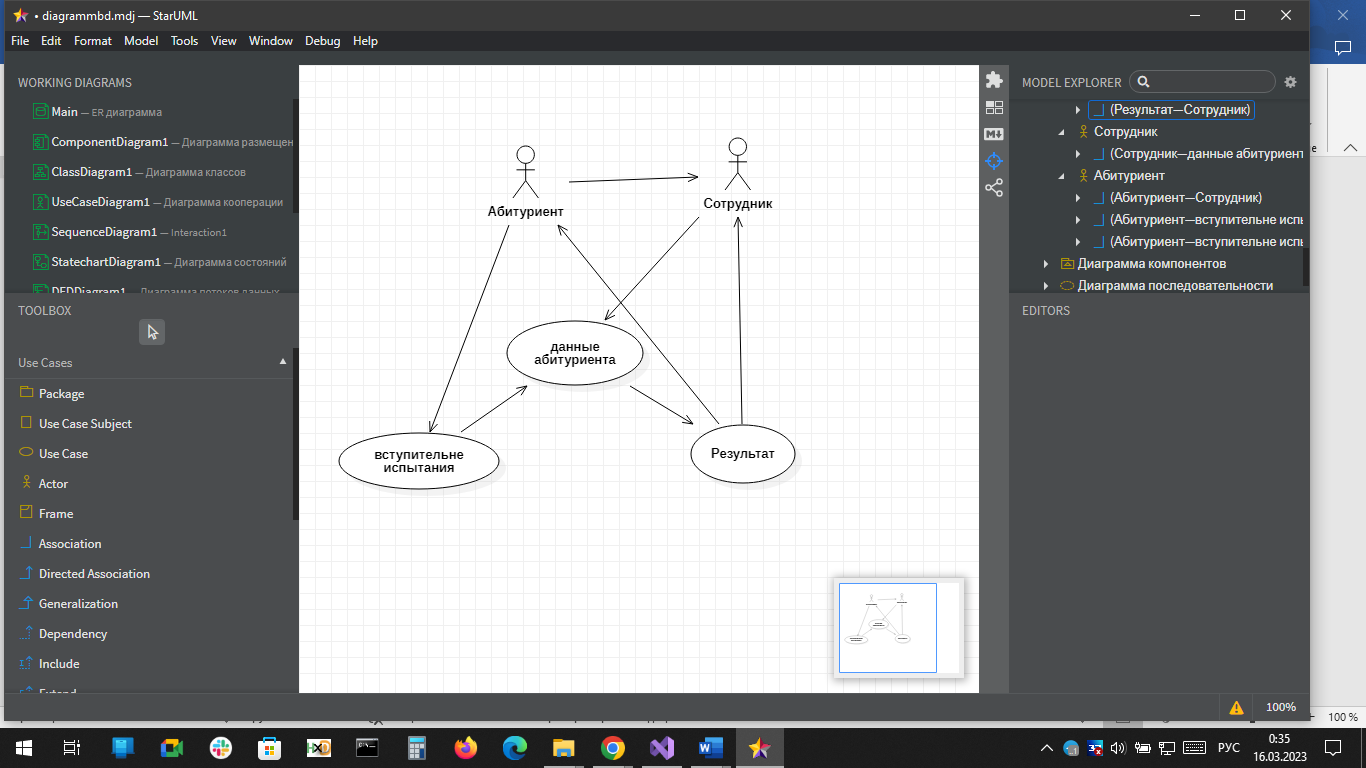


Рисунок 16 –Диаграммы кооперации

## 4.5 Диаграмма размещения

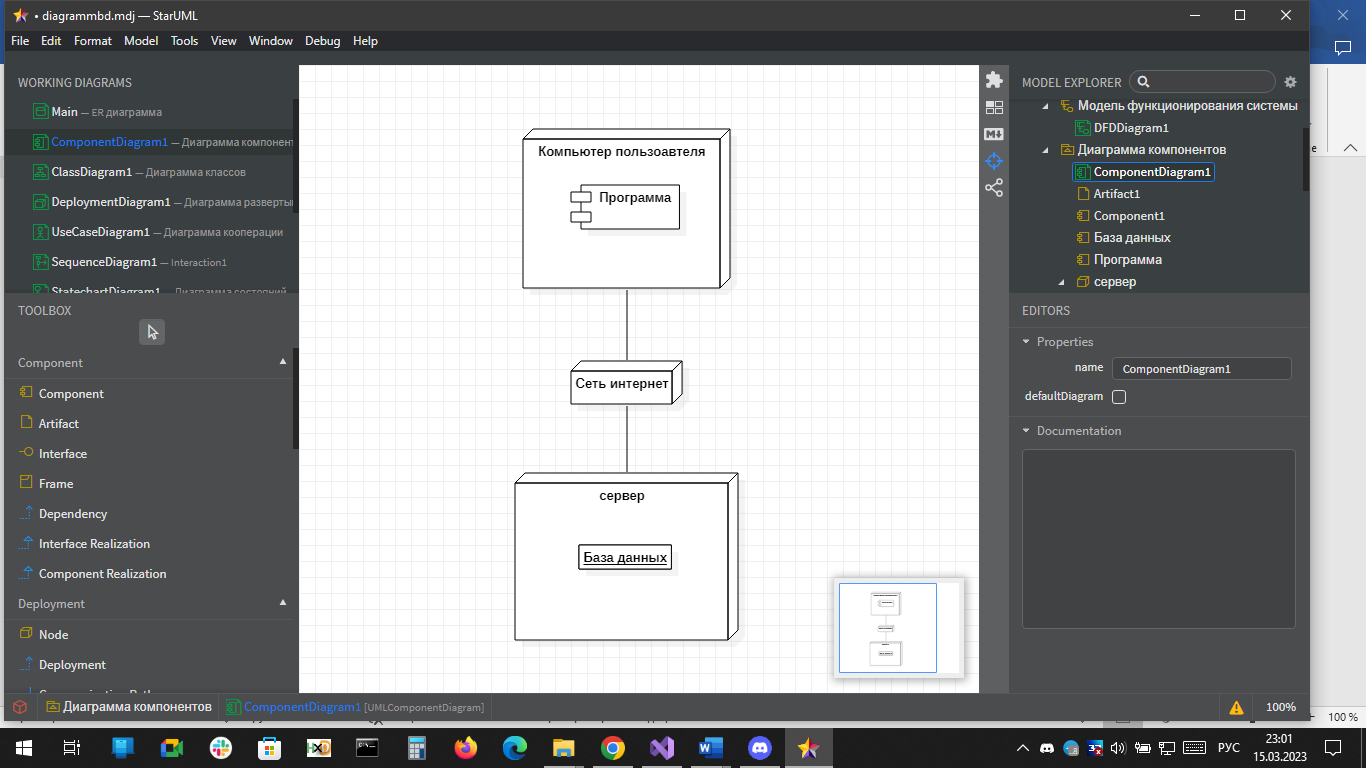


Рисунок 17 –Диаграмма размещения

## 4.6 Диаграмма состояний

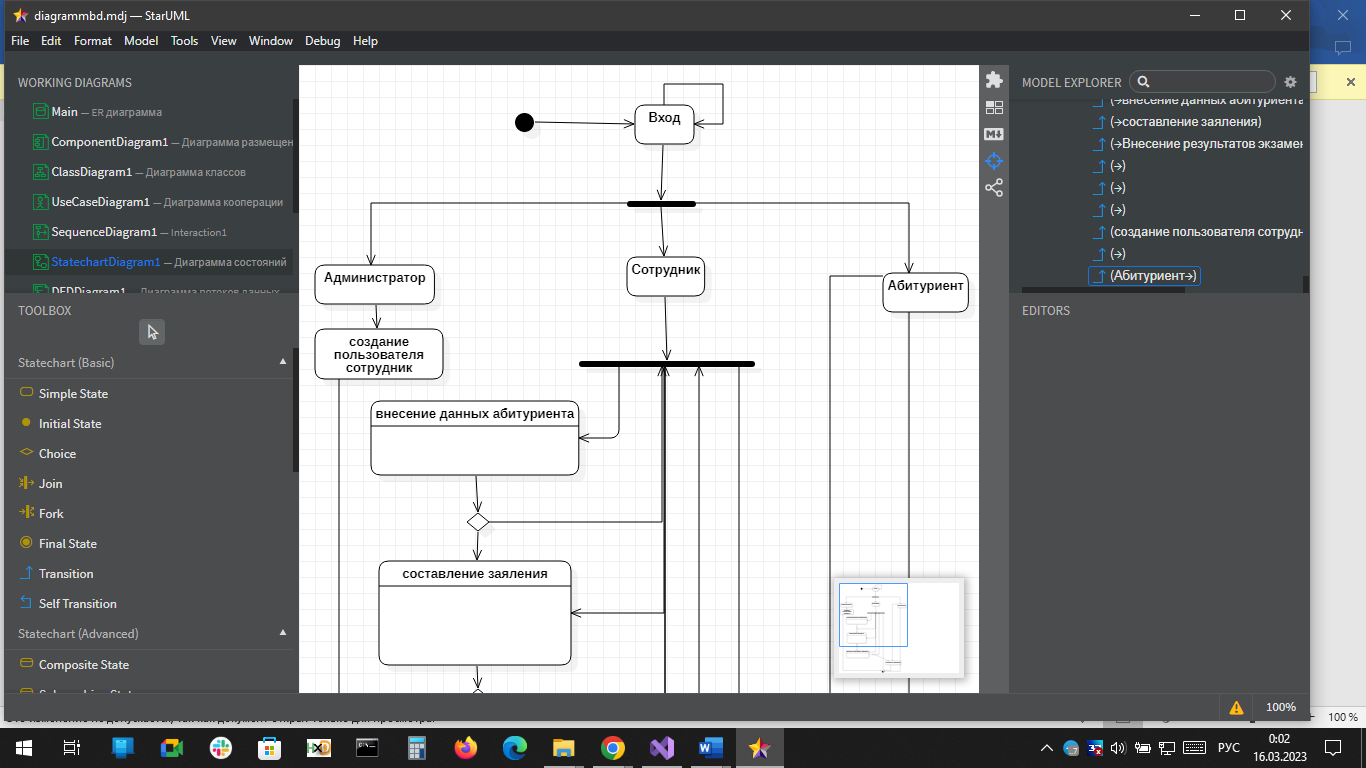


Рисунок 18 –Диаграмма состояний

## 4.7 Диаграммы потоков данных

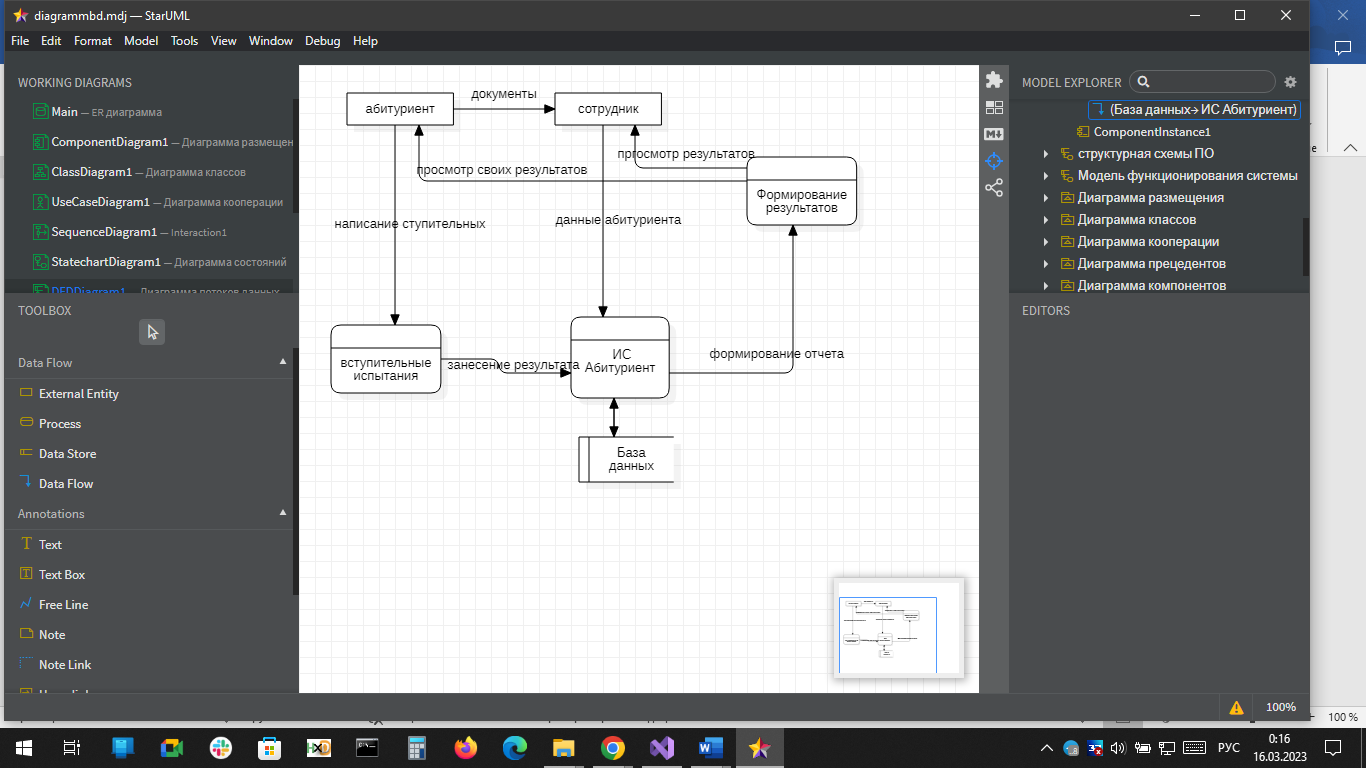


Рисунок 19 –Диаграммы потоков данных

## 4.8 Диаграмма классов

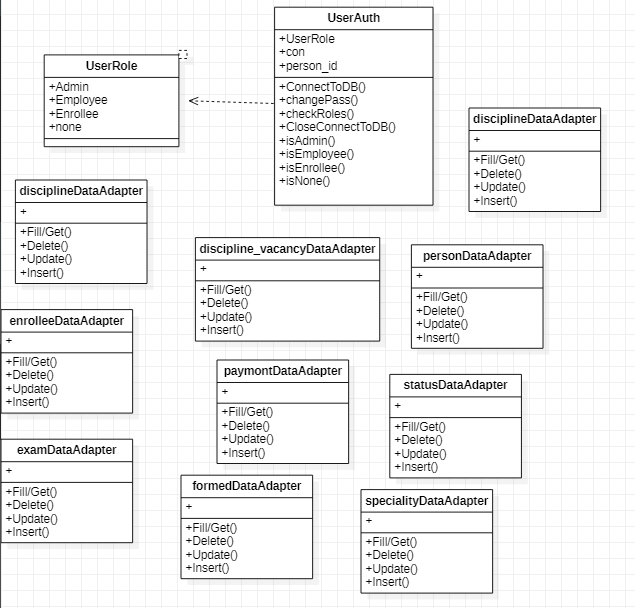


Рисунок 20 –Диаграмма классов

## 4.9 Диаграмма последовательности

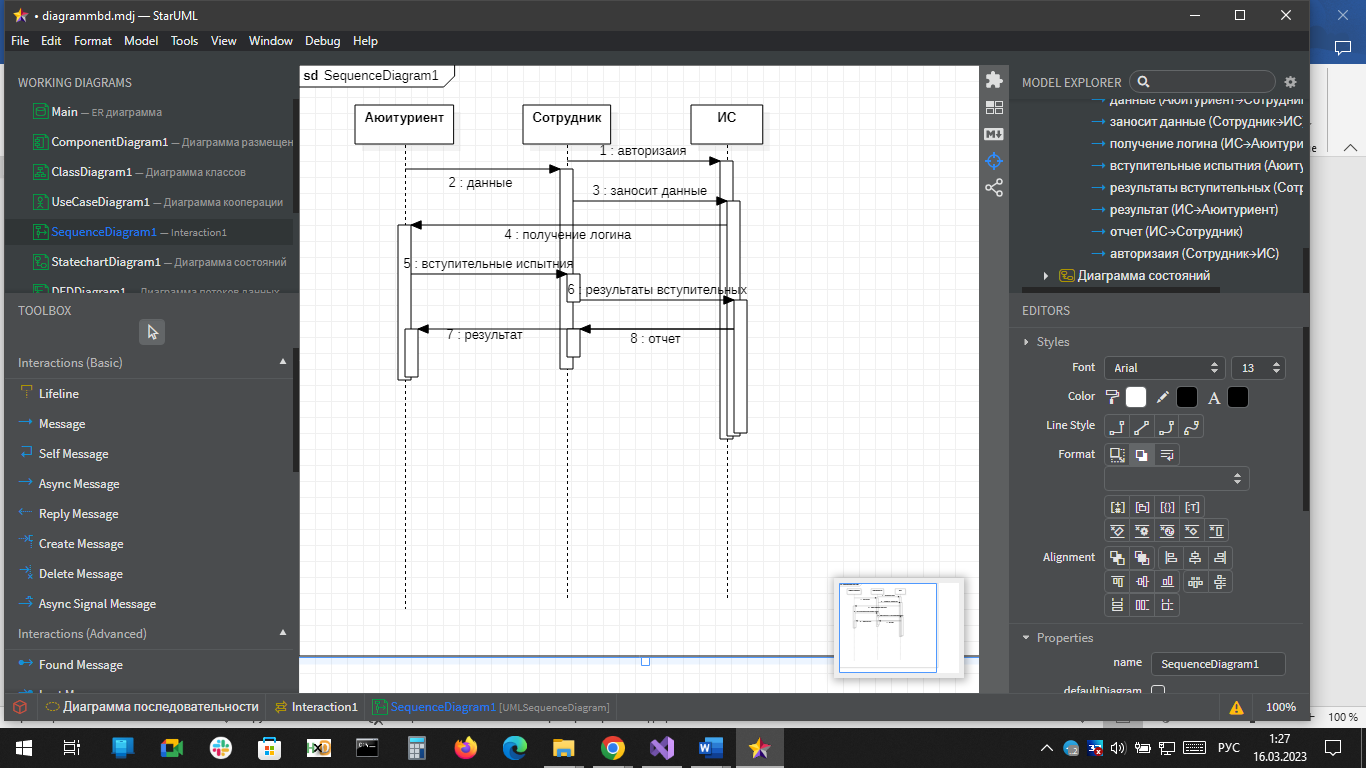


Рисунок 21 –Диаграмма последовательности

## 4.10 Диаграмма прецедентов

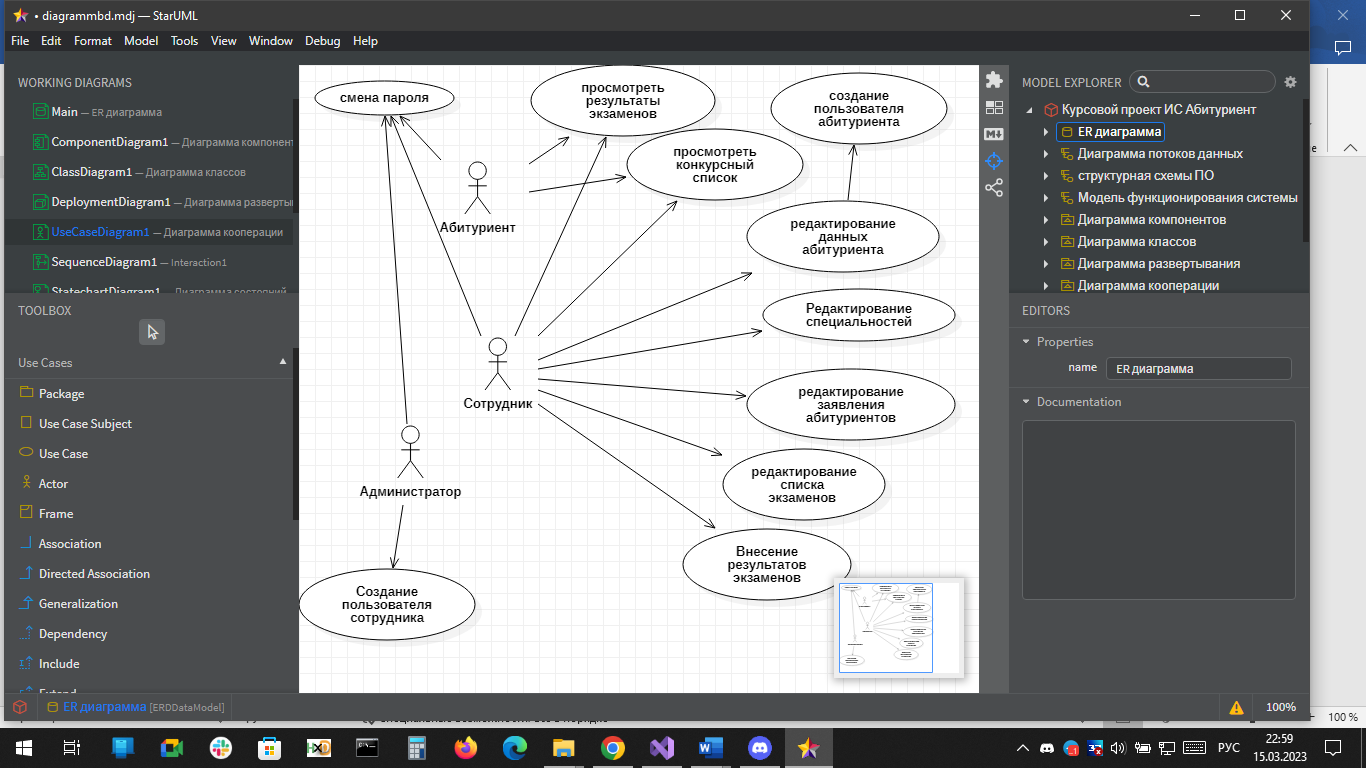


Рисунок 22 –Диаграмма прецедентов

# 5 Реализация приложения базы данных

## 5.1 Выбор среды программирования

В процессе разработки приложения будет использоваться среда Microsoft Visual Studio 2022, поддерживающая различные языки программирования и интегрированную систему управления сторонними пакетами NuGet для платформ разработки Microsoft. Для системы контроля версий выбрана распределенная система Git, созданная Линусом Торвальдсом для управления разработкой ядра Linux в 2005 году. .NET Foundation отвечает за управление .NET Framework и другими связанными технологиями.

## 5.2 Схема меню приложения

Меню приложения различается для разных ролей, на рисунке 22 представлено меню для роли администратор.

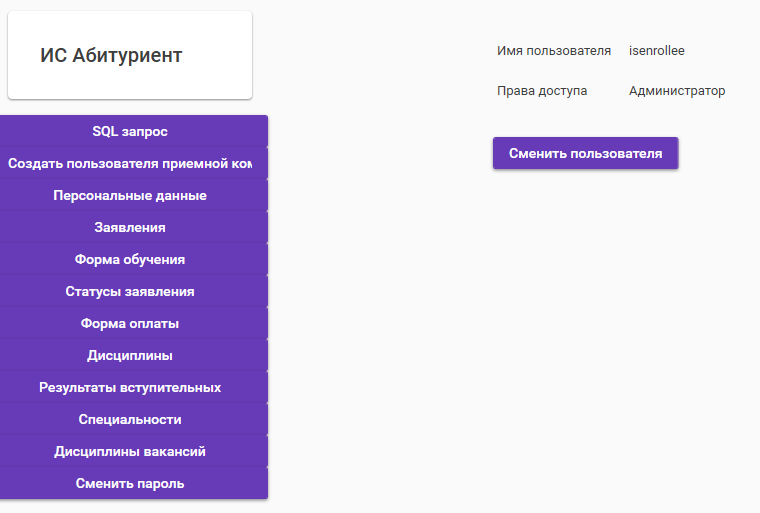


Рисунок 23 –Меню администратора

На рисунке 24 представлено меню для сотрудников.

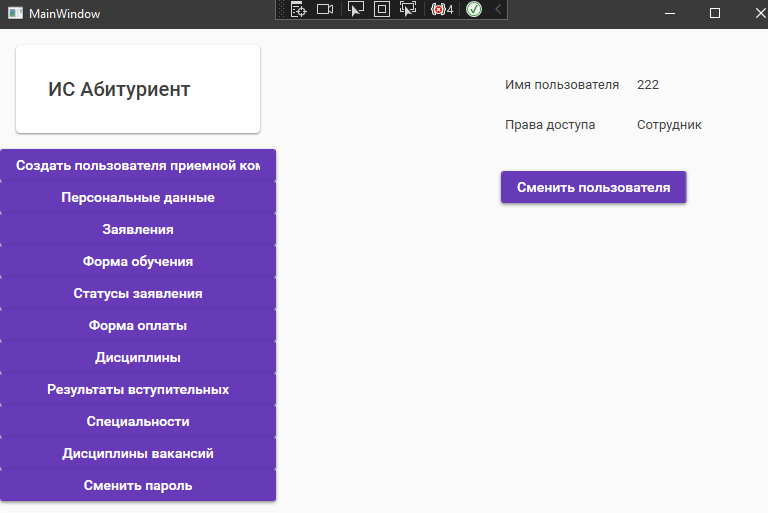


Рисунок 24 –Меню сотрудника

На рисунке 25 представлено меню для абитуриентов.

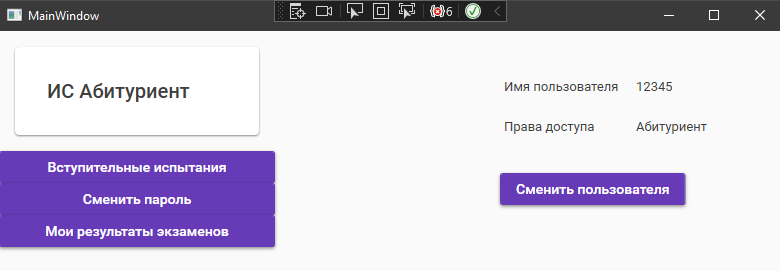


Рисунок 25 –Меню абитуриента

## 5.3 Окна приложения или руководство пользователя

После запуска мы видим окно авторизации, рисунок 26.

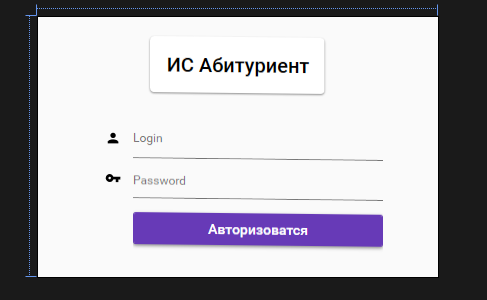


Рисунок 26 – Окно авторизации

После первого запуска, администратору необходимо создать пользователей для сотрудников, рисунок 27.

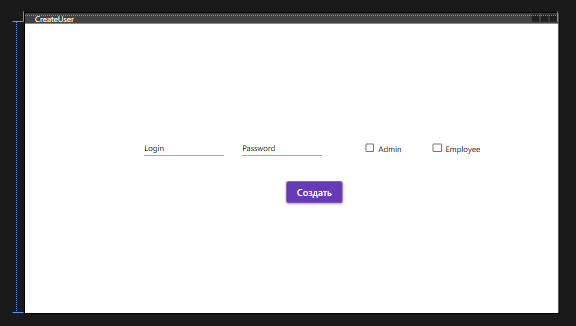


Рисунок 27 –Окно создания пользователя

Пользователи смогут сменить свой пароль, созданный администратором, рисунок 28.

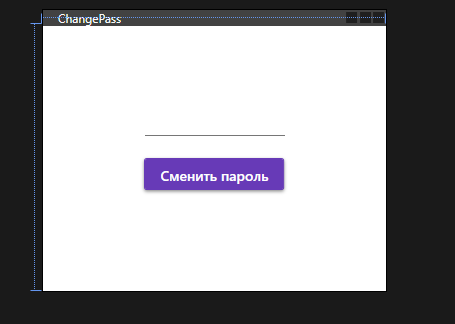


Рисунок 28 –Окно смены пароля

Сотрудник для начала вносит данные абитуриента, показано на рисунке 29. И при сохранении, создается пользователь абитуриента, с логином и паролем, его номера СНИЛС.

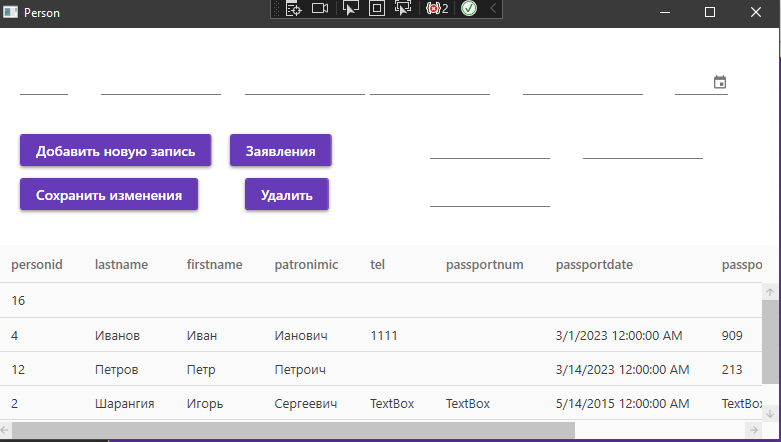


Рисунок 29 –Окно персональных данных

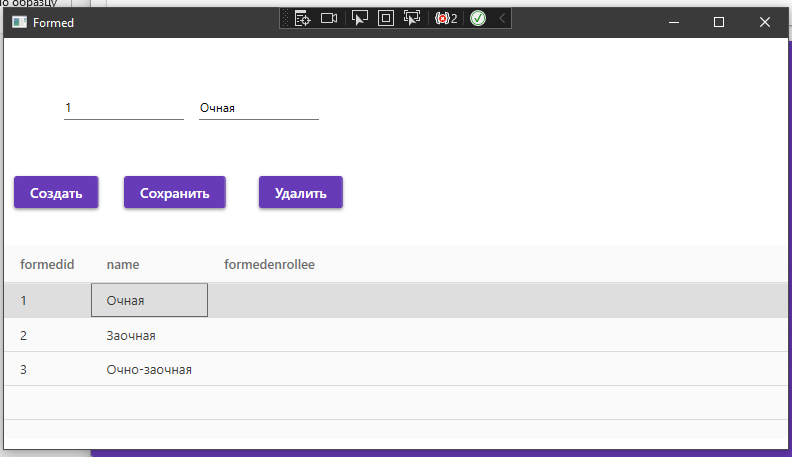


Рисунок 30 –Окно форма обучения

После добавления всех данных, сотрудник составляет заявление абитуриента, рисунок 31.

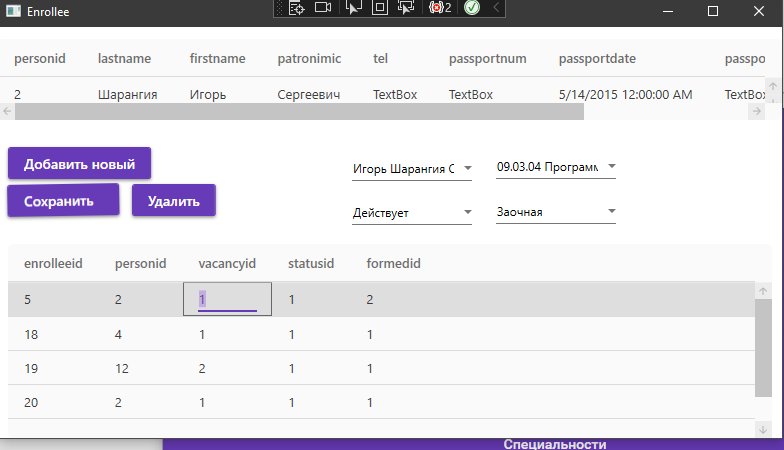


Рисунок 31 –Окно персональные данные

На списке всех вакантных мест показано на рисунке 32, есть возможность просмотра отчета результатов вступительных испытаний, показано на рисунке 33.

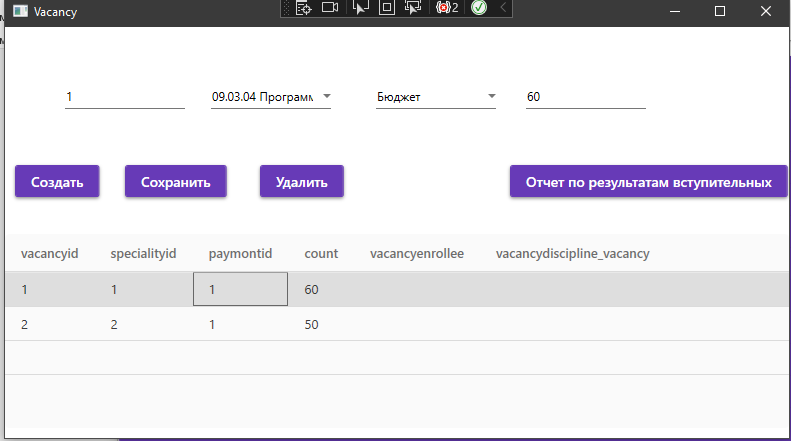


Рисунок 32 –Окно вакансий

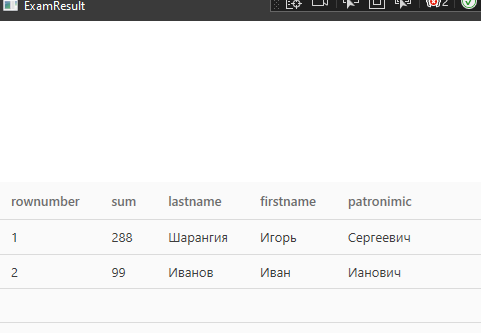


Рисунок 33 –Окно результатов вступительных испытаний

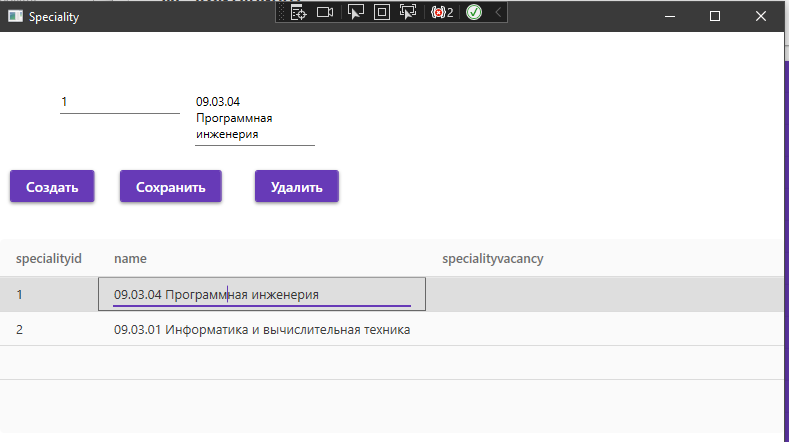


Рисунок 34 –Окно специальности

Абитуриент может просмотреть оценки своих вступительных испытаний, рисунок 35.

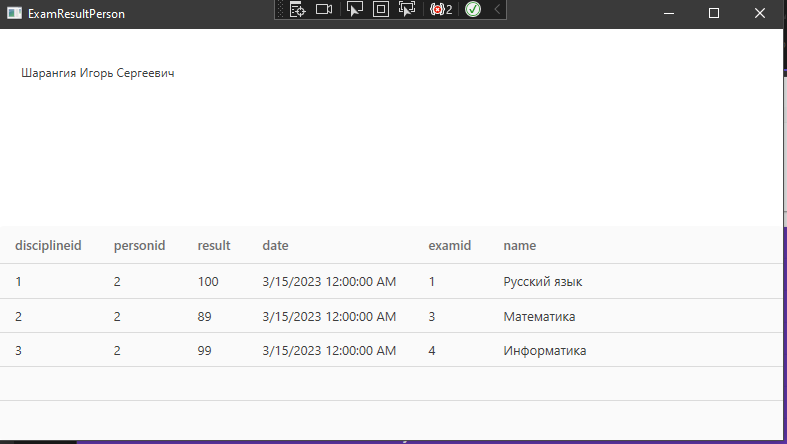


Рисунок 35 –Окно результаты вступительных

## 5.4 Запросы к базе данных

Листинг 1 Запрос вакансии по person\_id

SELECT enrollee\_id, formed\_id, person\_id, status\_id, vacancy\_id FROM enrollee WHERE (person\_id = @person\_id)

Запрос рейтинга вступительных испытаний по vacancy\_id

SELECT row\_number() OVER(), sum(exam.result),person.lastname, person.firstname,person.patronimic

FROM discipline\_vacancy INNER JOIN

vacancy ON discipline\_vacancy.vacancy\_id = vacancy.vacancy\_id

INNER JOIN

exam ON exam.discipline\_id = discipline\_vacancy.discipline\_id

INNER JOIN

person ON person.person\_id = exam.person\_id

INNER JOIN

enrollee ON enrollee.person\_id = exam.person\_id AND enrollee.vacancy\_id = vacancy.vacancy\_id

WHERE (vacancy.vacancy\_id = @vacancy\_id AND enrollee.formed\_id = @formed\_id AND enrollee.status\_id = 1)

GROUP BY person.person\_id

ORDER BY sum DESC

## 5.5 Функции и процедуры обработки данных

Хранимые функции для создания пользователя абитуриента при сохранении данных и при изменении представлены в листинге 2 и листинге 3.

Листинг 2 функции создания пользователя при сохранении данных

DECLARE

BEGIN

IF (NEW.snils IS NOT NULL) THEN

EXECUTE FORMAT('CREATE USER %I WITH PASSWORD %L IN GROUP enrollee', NEW.snils, NEW.snils);

END IF;

RETURN NEW;

END

Листинг 3 функции создания пользователя при сохранении данных при изменении

DECLARE

BEGIN

IF (NEW.snils <> OLD.snils OR (OLD.snils IS NULL AND NEW.snils IS NOT NULL)) THEN

EXECUTE FORMAT('CREATE USER %I WITH PASSWORD %L IN GROUP enrollee', NEW.snils, NEW.snils);

END IF;

RETURN NEW;

END

## 5.6 Формирование отчетной документации

Листинг 4 Формирование отчета с результатами экзамена

var dialog = new Microsoft.Win32.SaveFileDialog();

dialog.FileName = "Отчет"; // Default file name

dialog.DefaultExt = ".docx"; // Default file extension

dialog.Filter = "Word documents (.docx)|\*.docx"; // Filter files by extension

// Show open file dialog box

bool? result = dialog.ShowDialog();

// Process open file dialog box results

if (result == true)

{

var outfile = File.Open(dialog.FileName, FileMode.OpenOrCreate);

using (var doc = Configuration.Factory.Open(File.Open("examperson.docx", FileMode.Open), "docx", outfile))

{

doc.Process(

new

{

table1 = dataTable.exam,

firstname = dataTable.person.Rows[0]["firstname"],

lastname = dataTable.person.Rows[0]["lastname"],

patronomic = dataTable.person.Rows[0]["patronimic"]

});

}

outfile.Close();

}

}

Результат показан на рисунке 36.

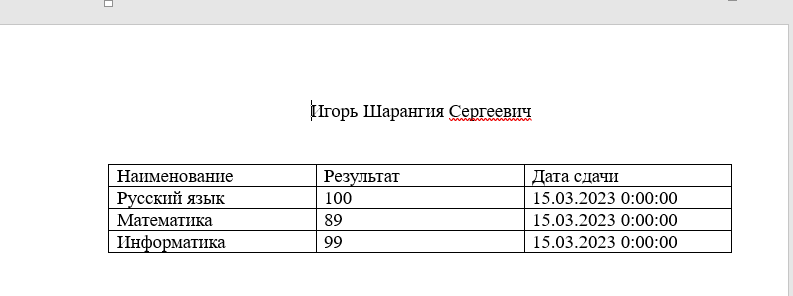


Рисунок 36 – Результат вступительных

Листинг 5 Формирование отчета конкурса

var dialog = new Microsoft.Win32.SaveFileDialog();

dialog.FileName = "Отчет"; // Default file name

dialog.DefaultExt = ".docx"; // Default file extension

dialog.Filter = "Word documents (.docx)|\*.docx"; // Filter files by extension

// Show open file dialog box

bool? result = dialog.ShowDialog();

// Process open file dialog box results

if (result == true)

{

var outfile = File.Open(dialog.FileName, FileMode.OpenOrCreate);

using (var doc = Configuration.Factory.Open(File.Open("Concurs.docx", FileMode.Open), "docx", outfile))

{

doc.Process(

new

{

Table1 = dataTable.vacancyresult,

specname = dataTable.speciality.Rows[0]["name"],

speccount = dataTable.vacancy.Rows[0]["count"]

});

}

outfile.Close();

}

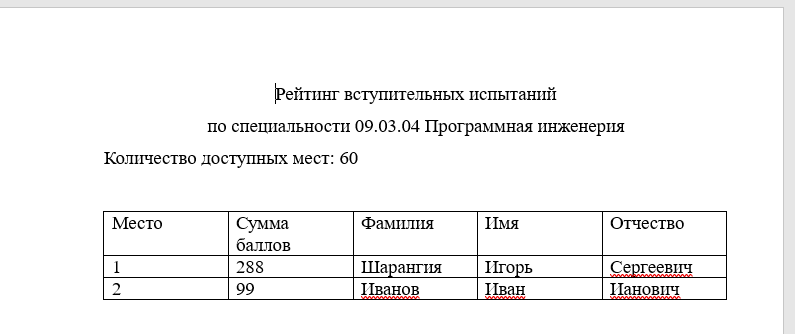


Рисунок 37 –Отчет рейтинг вступительных

# 6 Тестирование программного продукта

Листинг 6 Тесты для роли админ

internal class Admin

{

UserAuth auth;

isenrolleeDataSet isenrolleeDataSet;

[SetUp]

public void Setup()

{

auth = new UserAuth("AdminTest", "AdminTest");

UserAuth.getUserAuth();

isenrolleeDataSet = new isenrolleeDataSet();

}

[Test]

public void Test1()

{

Assert.IsTrue(auth.ConnectToDB(), "Нет подключения к бд");

}

[Test]

public void Test2()

{

auth.ConnectToDB();

Assert.IsTrue(UserAuth.isAdmin, "Роль пользователя не Администратор!");

}…}

Листинг 7 Тесты для роли Сотрудник

internal class Employee

{

UserAuth auth;

isenrolleeDataSet isenrolleeDataSet;

[SetUp]

public void Setup()

{

auth = new UserAuth("EmployeeTest", "EmployeeTest");

UserAuth.getUserAuth();

isenrolleeDataSet = new isenrolleeDataSet();

}

[Test]

public void Test1()

{

Assert.IsTrue(auth.ConnectToDB(), "Нет подключения к бд");

}

[Test]

public void Test2()

{

auth.ConnectToDB();

Assert.IsTrue(UserAuth.isEmployee, "Роль пользователя не Сотрудник!");

} …}

Листинг 8 Тесты для роли абитуриент

internal class Enrollee

{

UserAuth auth;

isenrolleeDataSet isenrolleeDataSet;

[SetUp]

public void Setup()

{

auth = new UserAuth("EnrolleeTest", "EnrolleeTest");

UserAuth.getUserAuth();

isenrolleeDataSet = new isenrolleeDataSet();

}

[Test]

public void Test1()

{

Assert.IsTrue(auth.ConnectToDB(), "Нет подключения к бд");

}

public void Test13()

{

try

{

auth.ConnectToDB();

personTableAdapter DataAdapter;

DataAdapter = new personTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.newPersonRow();

// обноляем записи в таблице datagrid

DataAdapter.Fill(isenrolleeDataSet.person);

}

catch

{

Assert.Pass();

}

Assert.Fail("У пользователя есть права на запись в таблицу person");

}…}

Полный текст тестов в приложении, результат тестирования показан на рисунке 38.

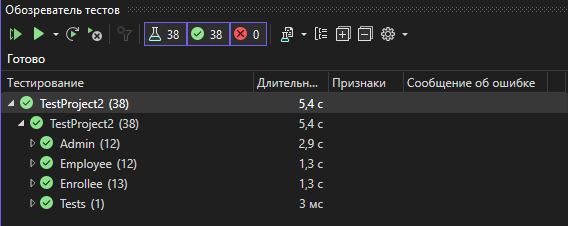


Рисунок 38 –Выполнение тестов

# Заключение

В заключение, можно сказать, что информационная система абитуриент является очень важным инструментом, который позволяет повысить эффективность и качество работы приемной комиссии. Она позволяет автоматизировать множество процессов, связанных с приемом и обработкой заявлений на поступление, упростить и ускорить работу персонала, уменьшить количество ошибок и несоответствий при оформлении документов.

В разработке информационной системы абитуриент необходимо учитывать множество факторов, таких как целевая аудитория, требования заказчика, стандарты безопасности и др. Важно также обеспечить удобство использования системы для всех пользователей, даже для тех, кто ранее не имел опыта работы с компьютерами.

Информационная система абитуриент не только упрощает и ускоряет работу персонала, но также облегчает процесс поступления для абитуриентов. Она позволяет уменьшить время и затраты на подачу заявления, а также предоставляет возможность получить информацию о своем статусе в режиме реального времени.

Таким образом, разработка информационной системы абитуриент – это актуальная и практически важная задача, которая требует комплексного и профессионального подхода. Результаты этой работы могут оказать положительное влияние на работу приемной комиссии и на процесс поступления в образовательное учреждение в целом.

# Список использованных источников

1. Албахари, Джозеф C# 6.0. Справочник / Джозеф Албахари , Бен Албахари. – Изд. Вильямс, 2018. - 1040 c.
2. StarUML documentation [Электронный ресурс]. URL: <https://docs.staruml.io/user-guide/readme> (дата обращения 6.03.2023).
3. PostgreSQL in C# .NET with Npgsql, Dapper, and Entity Framework: The Complete Guide [Электронный ресурс]. URL: <https://michaelscodingspot.com/postgres-in-csharp/> (дата обращения 20.02.2023).
4. Class NpgsqlDataAdapter [Электронный ресурс]. URL: <https://www.npgsql.org/doc/api/Npgsql.NpgsqlDataAdapter.html> (дата обращения 22.02.2023).
5. ИСПОЛЬЗОВАНИЕ РОЛЕЙ И УПРАВЛЕНИЕ ДОСТУПОМ В POSTGRESQL [Электронный ресурс]. URL: <https://www.8host.com/blog/ispolzovanie-rolej-i-upravlenie-dostupom-v-postgresql/> (дата обращения 24.02.2023).
6. SqlDataAdapter.UpdateCommand [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.data.sqlclient.sqldataadapter.updatecommand> (дата обращения 8.03.2023).
7. Templater DOCUMENTS [Электронный ресурс]. URL: <https://templater.info/documents> (дата обращения 3.03.2023).

# Приложение А Текст программы

public partial class ChangePass : Window

{

public ChangePass()

{

InitializeComponent();

}

private void button\_Click(object sender, RoutedEventArgs e)

{

if (UserAuth.getUserAuth().changePass(textBox1.Text))

{

MessageBox.Show("Пароль успешно сменен");

Close();

}

else

{

MessageBox.Show("Не удалось сменить пароль");

}

}

public partial class CreateUser : Window

{

public CreateUser()

{

InitializeComponent();

}

private void button\_Click(object sender, RoutedEventArgs e)

{

UserAuth con = UserAuth.getUserAuth();

try

{

var roles = "";

if ((checkBox.IsChecked ?? true) || (checkBox.IsChecked ?? true))

{

roles = "IN GROUP";

}

var role = "";

if (checkBox.IsChecked ?? true)

{

role = "admin";

}

if (checkBox1.IsChecked ?? true)

{

role = "employee";

}

string query = String.Format("CREATE USER \"{0}\" WITH PASSWORD '{1}' {2} {3}",

textBox.Text,

textBox1.Text,

roles,

role

);

var cmd = new NpgsqlCommand(query, con.con);

MessageBox.Show("Пользователь создан");

}

catch (NpgsqlException ex)

{

MessageBox.Show(ex.ToString());

}

}

}

public partial class Discipline : Window

{

isenrolleeDataSet dataTable;

disciplineTableAdapter disciplineDataAdapter;

public RolesViewModel ViewModel { get; set; }

public Discipline()

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

disciplineDataAdapter = new disciplineTableAdapter();

disciplineDataAdapter.Connection = con.con;

dataTable = new isenrolleeDataSet();

disciplineDataAdapter.Fill(dataTable.discipline);

dataGrid.ItemsSource = dataTable.discipline.DefaultView;

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

/\* Сохранение \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

selectedItem[0] = textBox.Text;

selectedItem[1] = textBox1.Text;

}

disciplineDataAdapter.Adapter.UpdateCommand = new NpgsqlCommandBuilder(disciplineDataAdapter.Adapter).GetUpdateCommand();

disciplineDataAdapter.Adapter.Update(dataTable.discipline);

}

private void dataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

/\* Заполнение полей по клику \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

textBox.Text = selectedItem[0].ToString();

textBox1.Text = selectedItem[1].ToString();

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

/\* Создание \*/

disciplineDataAdapter.Adapter.InsertCommand = new NpgsqlCommandBuilder(disciplineDataAdapter.Adapter).GetInsertCommand();

disciplineDataAdapter.InsertQuery1();

// обноляем записи в таблице datagrid

disciplineDataAdapter.Fill(dataTable.discipline);

// выбираем последнюю запись, это будет та что создали

dataGrid.SelectedIndex = dataTable.discipline.Count - 1;

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

/\* Удаление \*/

int index = dataGrid.SelectedIndex;

var selectedItem = dataGrid.SelectedItem;

if (selectedItem != null)

{

dataTable.discipline.DefaultView.Delete(index);

}

disciplineDataAdapter.Adapter.DeleteCommand = new NpgsqlCommandBuilder(disciplineDataAdapter.Adapter).GetDeleteCommand();

disciplineDataAdapter.Adapter.Update(dataTable.discipline);

}

}

public partial class Discipline\_vacancy : Window

{

isenrolleeDataSet dataTable;

isenrolleeDataSet datasetCombobox;

discipline\_vacancyTableAdapter discipline\_vacancyTableDataAdapter;

disciplineTableAdapter disciplineComboboxData;

vacancyTableAdapter vacancyComboboxData;

public RolesViewModel ViewModel { get; set; }

public Discipline\_vacancy()

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

discipline\_vacancyTableDataAdapter = new discipline\_vacancyTableAdapter();

discipline\_vacancyTableDataAdapter.Connection = con.con;

dataTable = new isenrolleeDataSet();

discipline\_vacancyTableDataAdapter.Fill(dataTable.discipline\_vacancy);

dataGrid.ItemsSource = dataTable.discipline\_vacancy.DefaultView;

datasetCombobox = new isenrolleeDataSet();

disciplineComboboxData = new disciplineTableAdapter();

disciplineComboboxData.Connection = con.con;

disciplineComboboxData.Fill(datasetCombobox.discipline);

comboBox.ItemsSource = datasetCombobox.discipline.DefaultView;

vacancyComboboxData = new vacancyTableAdapter();

vacancyComboboxData.Connection = con.con;

vacancyComboboxData.Fill(datasetCombobox.vacancy);

comboBox1.ItemsSource = datasetCombobox.vacancy.DefaultView;

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

/\* Сохранение \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

selectedItem[2] = textBox.Text;

selectedItem[0] = comboBox.SelectedValue;

selectedItem[1] = comboBox1.SelectedValue;

}

discipline\_vacancyTableDataAdapter.Adapter.UpdateCommand = new NpgsqlCommandBuilder(discipline\_vacancyTableDataAdapter.Adapter).GetUpdateCommand();

discipline\_vacancyTableDataAdapter.Adapter.Update(dataTable.discipline\_vacancy);

}

private void dataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

/\* Заполнение полей по клику \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

textBox.Text = selectedItem[2].ToString();

comboBox.SelectedValue = selectedItem[0];

comboBox1.SelectedValue = selectedItem[1];

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

/\* Создание \*/

discipline\_vacancyTableDataAdapter.Adapter.InsertCommand = new NpgsqlCommandBuilder(discipline\_vacancyTableDataAdapter.Adapter).GetInsertCommand();

discipline\_vacancyTableDataAdapter.InsertQuery1();

// обноляем записи в таблице datagrid

discipline\_vacancyTableDataAdapter.Fill(dataTable.discipline\_vacancy);

// выбираем последнюю запись, это будет та что создали

dataGrid.SelectedIndex = dataTable.discipline\_vacancy.Count - 1;

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

/\* Удаление \*/

int index = dataGrid.SelectedIndex;

var selectedItem = dataGrid.SelectedItem;

if (selectedItem != null)

{

dataTable.discipline\_vacancy.DefaultView.Delete(index);

}

discipline\_vacancyTableDataAdapter.Adapter.DeleteCommand = new NpgsqlCommandBuilder(discipline\_vacancyTableDataAdapter.Adapter).GetDeleteCommand();

discipline\_vacancyTableDataAdapter.Adapter.Update(dataTable.discipline\_vacancy);

}

}

public partial class Enrollee : Window

{

enrolleeTableAdapter enrolleeDataAdapter;

isenrolleeDataSet dataTable;

isenrolleeDataSet datasetCombobox;

personTableAdapter cmdperson;

personTableAdapter personComboboxData;

//bonusTableAdapter bonusComboboxData;

vacancy1TableAdapter vacancy1ComboboxData;

statusTableAdapter statusComboboxData;

formedTableAdapter formedComboboxData;

int person\_id;

public RolesViewModel ViewModel { get; set; }

public Enrollee(int \_person\_id = 0)

{

person\_id = \_person\_id;

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

try

{

enrolleeDataAdapter = new enrolleeTableAdapter();

enrolleeDataAdapter.Connection = con.con;

cmdperson = new personTableAdapter();

cmdperson.Connection = con.con;

dataTable = new isenrolleeDataSet();

if (person\_id > 0)

{

cmdperson.FillByPersonid(dataTable.person, person\_id);

enrolleeDataAdapter.FillByPersonId(dataTable.enrollee, person\_id);

}

else

{

enrolleeDataAdapter.Fill(dataTable.enrollee);

}

datasetCombobox = new isenrolleeDataSet();

// bonusComboboxData = new bonusTableAdapter();

//bonusComboboxData.Connection = con.con;

personComboboxData = new personTableAdapter();

personComboboxData.Connection = con.con;

vacancy1ComboboxData = new vacancy1TableAdapter();

vacancy1ComboboxData.Connection = con.con;

statusComboboxData = new statusTableAdapter();

statusComboboxData.Connection = con.con;

formedComboboxData = new formedTableAdapter();

formedComboboxData.Connection = con.con;

//bonusComboboxData.Fill(datasetCombobox.bonus);

personComboboxData.Fill(datasetCombobox.person);

vacancy1ComboboxData.Fill(datasetCombobox.vacancy1);

statusComboboxData.Fill(datasetCombobox.status);

formedComboboxData.Fill(datasetCombobox.formed);

//comboBox.ItemsSource = datasetCombobox.bonus.DefaultView;

comboBox1.ItemsSource = datasetCombobox.person.DefaultView;

comboBox2.ItemsSource = datasetCombobox.vacancy1.DefaultView;

comboBox3.ItemsSource = datasetCombobox.status.DefaultView;

comboBox4.ItemsSource = datasetCombobox.formed.DefaultView;

dataGrid.ItemsSource = dataTable.enrollee.DefaultView;

dataGrid1.ItemsSource = dataTable.person.DefaultView;

}

catch (NpgsqlException ex)

{

MessageBox.Show(ex.ToString());

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

cmdperson.FillByPersonid(dataTable.person, selectedItem[1]);

//selectedItem[1] = comboBox.SelectedValue;

selectedItem[1] = comboBox1.SelectedValue;

selectedItem[2] = comboBox2.SelectedValue;

selectedItem[3] = comboBox3.SelectedValue;

selectedItem[4] = comboBox4.SelectedValue;

}

enrolleeDataAdapter.Adapter.UpdateCommand = new NpgsqlCommandBuilder(enrolleeDataAdapter.Adapter).GetUpdateCommand();

enrolleeDataAdapter.Adapter.Update(dataTable.enrollee);

}

private void dataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

cmdperson.FillByPersonid(dataTable.person, selectedItem[1]);

//comboBox.SelectedValue = selectedItem[1];

comboBox1.SelectedValue = selectedItem[1];

comboBox2.SelectedValue = selectedItem[2];

comboBox3.SelectedValue = selectedItem[3];

comboBox4.SelectedValue = selectedItem[4];

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

int index = dataGrid.SelectedIndex;

var selectedItem = dataGrid.SelectedItem;

if (selectedItem != null)

{

dataTable.enrollee.DefaultView.Delete(index);

}

enrolleeDataAdapter.Adapter.DeleteCommand = new NpgsqlCommandBuilder(enrolleeDataAdapter.Adapter).GetDeleteCommand();

enrolleeDataAdapter.Adapter.Update(dataTable.enrollee);

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

if (person\_id > 0)

{

// создаем запись в бд

enrolleeDataAdapter.InsertQuery1( person\_id);

}

else

{

// создаем запись в бд

enrolleeDataAdapter.InsertQuery2();

}

enrolleeDataAdapter.Adapter.InsertCommand = new NpgsqlCommandBuilder(enrolleeDataAdapter.Adapter).GetInsertCommand();

// обноляем записи в таблице datagrid

enrolleeDataAdapter.Fill(dataTable.enrollee);

// выбираем последнюю запись, это будет та что создали

dataGrid.SelectedIndex = dataTable.enrollee.Count - 1;

}

}

public partial class Exam : Window

{

isenrolleeDataSet dataTable;

examTableAdapter examDataAdapter;

isenrolleeDataSet datasetCombobox;

disciplineTableAdapter disciplineComboboxData;

personTableAdapter personComboboxData;

public RolesViewModel ViewModel { get; set; }

public Exam()

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

examDataAdapter = new examTableAdapter();

examDataAdapter.Connection = con.con;

dataTable = new isenrolleeDataSet();

examDataAdapter.Fill(dataTable.exam);

dataGrid.ItemsSource = dataTable.exam.DefaultView;

datasetCombobox = new isenrolleeDataSet();

disciplineComboboxData = new disciplineTableAdapter();

disciplineComboboxData.Connection = con.con;

disciplineComboboxData.Fill(datasetCombobox.discipline);

comboBox.ItemsSource = datasetCombobox.discipline.DefaultView;

personComboboxData = new personTableAdapter();

personComboboxData.Connection = con.con;

personComboboxData.Fill(datasetCombobox.person);

comboBox1.ItemsSource = datasetCombobox.person.DefaultView;

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

/\* Сохранение \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

selectedItem[0] = comboBox.SelectedValue;

selectedItem[1] = comboBox1.SelectedValue;

selectedItem[2] = textBox.Text;

selectedItem[3] = datePicker.SelectedDate;

selectedItem[4] = textBox1.Text;

}

examDataAdapter.Adapter.UpdateCommand = new NpgsqlCommandBuilder(examDataAdapter.Adapter).GetUpdateCommand();

examDataAdapter.Adapter.Update(dataTable.exam);

}

private void dataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

/\* Заполнение полей по клику \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

comboBox.SelectedValue = selectedItem[0];

comboBox1.SelectedValue = selectedItem[1];

textBox.Text = selectedItem[2].ToString();

datePicker.Text = selectedItem[3].ToString();

textBox1.Text = selectedItem[4].ToString();

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

/\* Создание \*/

examDataAdapter.Adapter.InsertCommand = new NpgsqlCommandBuilder(examDataAdapter.Adapter).GetInsertCommand();

examDataAdapter.InsertQuery1();

// обноляем записи в таблице datagrid

examDataAdapter.Fill(dataTable.exam);

// выбираем последнюю запись, это будет та что создали

dataGrid.SelectedIndex = dataTable.exam.Count - 1;

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

/\* Удаление \*/

int index = dataGrid.SelectedIndex;

var selectedItem = dataGrid.SelectedItem;

if (selectedItem != null)

{

dataTable.exam.DefaultView.Delete(index);

}

examDataAdapter.Adapter.DeleteCommand = new NpgsqlCommandBuilder(examDataAdapter.Adapter).GetDeleteCommand();

examDataAdapter.Adapter.Update(dataTable.exam);

}

}

public partial class ExamResult : Window

{

isenrolleeDataSet dataTable;

vacancyresultTableAdapter vacancyDataAdapter;

vacancyTableAdapter vacancytDataAdapter;

specialityTableAdapter specialitytDataAdapter;

public RolesViewModel ViewModel { get; set; }

public ExamResult(int vacancy\_id = 0, int formed\_id = 0)

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

vacancyDataAdapter = new vacancyresultTableAdapter();

vacancytDataAdapter = new vacancyTableAdapter();

specialitytDataAdapter = new specialityTableAdapter();

vacancyDataAdapter.Connection = con.con;

vacancytDataAdapter.Connection = con.con;

specialitytDataAdapter.Connection = con.con;

dataTable = new isenrolleeDataSet();

vacancyDataAdapter.Fill(dataTable.vacancyresult, vacancy\_id, formed\_id);

vacancytDataAdapter.FillBy2(dataTable.vacancy, vacancy\_id);

specialitytDataAdapter.FillBy(dataTable.speciality, Convert.ToInt32(dataTable.vacancy.Rows[0]["speciality\_id"]));

dataGrid.ItemsSource = dataTable.vacancyresult.DefaultView;

}

private void button\_Click(object sender, RoutedEventArgs e)

{

var dialog = new Microsoft.Win32.SaveFileDialog();

dialog.FileName = "Отчет"; // Default file name

dialog.DefaultExt = ".docx"; // Default file extension

dialog.Filter = "Word documents (.docx)|\*.docx"; // Filter files by extension

// Show open file dialog box

bool? result = dialog.ShowDialog();

// Process open file dialog box results

if (result == true)

{

var outfile = File.Open(dialog.FileName, FileMode.OpenOrCreate);

using (var doc = Configuration.Factory.Open(File.Open("Concurs.docx", FileMode.Open), "docx", outfile))

{

doc.Process(

new

{

Table1 = dataTable.vacancyresult,

specname = dataTable.speciality.Rows[0]["name"],

speccount = dataTable.vacancy.Rows[0]["count"]

});

}

outfile.Close();

}

}

}

public partial class ExamResultPerson : Window

{

isenrolleeDataSet dataTable;

examTableAdapter examDataAdapter;

isenrolleeDataSet datasetCombobox;

disciplineTableAdapter disciplineComboboxData;

personTableAdapter personComboboxData;

public RolesViewModel ViewModel { get; set; }

public ExamResultPerson(int person\_id = 0)

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

examDataAdapter = new examTableAdapter();

examDataAdapter.Connection = con.con;

personComboboxData = new personTableAdapter();

personComboboxData.Connection = con.con;

dataTable = new isenrolleeDataSet();

if (ViewModel.EnrolleeVisible)

{

person\_id = con.person\_id;

}

examDataAdapter.FillBy1(dataTable.exam, person\_id);

personComboboxData.FillByPersonid(dataTable.person, person\_id);

dataGrid.ItemsSource = dataTable.exam.DefaultView;

label.Content = dataTable.person.DefaultView[0].Row[1].ToString()+ " "+

dataTable.person.DefaultView[0].Row[2].ToString() + " " +

dataTable.person.DefaultView[0].Row[3].ToString();

}

private void button\_Click(object sender, RoutedEventArgs e)

{

var dialog = new Microsoft.Win32.SaveFileDialog();

dialog.FileName = "Отчет"; // Default file name

dialog.DefaultExt = ".docx"; // Default file extension

dialog.Filter = "Word documents (.docx)|\*.docx"; // Filter files by extension

// Show open file dialog box

bool? result = dialog.ShowDialog();

// Process open file dialog box results

if (result == true)

{

var outfile = File.Open(dialog.FileName, FileMode.OpenOrCreate);

using (var doc = Configuration.Factory.Open(File.Open("examperson.docx", FileMode.Open), "docx", outfile))

{

doc.Process(

new

{

table1 = dataTable.exam,

firstname = dataTable.person.Rows[0]["firstname"],

lastname = dataTable.person.Rows[0]["lastname"],

patronomic = dataTable.person.Rows[0]["patronimic"]

});

}

outfile.Close();

}

}

}

public partial class Formed : Window

{

isenrolleeDataSet dataTable;

formedTableAdapter formedDataAdapter;

public RolesViewModel ViewModel { get; set; }

public Formed()

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

formedDataAdapter = new formedTableAdapter();

formedDataAdapter.Connection = con.con;

dataTable = new isenrolleeDataSet();

formedDataAdapter.Fill(dataTable.formed);

dataGrid.ItemsSource = dataTable.formed.DefaultView;

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

/\* Сохранение \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

selectedItem[0] = textBox.Text;

selectedItem[1] = textBox1.Text;

}

formedDataAdapter.Adapter.UpdateCommand = new NpgsqlCommandBuilder(formedDataAdapter.Adapter).GetUpdateCommand();

formedDataAdapter.Adapter.Update(dataTable.formed);

}

private void dataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

/\* Заполнение полей по клику \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

textBox.Text = selectedItem[0].ToString();

textBox1.Text = selectedItem[1].ToString();

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

/\* Создание \*/

formedDataAdapter.Adapter.InsertCommand = new NpgsqlCommandBuilder(formedDataAdapter.Adapter).GetInsertCommand();

formedDataAdapter.InsertQuery1();

// обноляем записи в таблице datagrid

formedDataAdapter.Fill(dataTable.formed);

// выбираем последнюю запись, это будет та что создали

dataGrid.SelectedIndex = dataTable.formed.Count - 1;

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

/\* Удаление \*/

int index = dataGrid.SelectedIndex;

var selectedItem = dataGrid.SelectedItem;

if (selectedItem != null)

{

dataTable.formed.DefaultView.Delete(index);

}

formedDataAdapter.Adapter.DeleteCommand = new NpgsqlCommandBuilder(formedDataAdapter.Adapter).GetDeleteCommand();

formedDataAdapter.Adapter.Update(dataTable.formed);

}

}

public partial class LoginPage : Page

{

public LoginPage()

{

InitializeComponent();

}

private void button\_Click(object sender, RoutedEventArgs e)

{

UserAuth user = new UserAuth(LoginBox.Text, passwordBox.Password);

if (user.ConnectToDB())

{

MainWindow mainwindow = new MainWindow();

mainwindow.Show();

Window.GetWindow(this).Close();

}

else

{

MessageBox.Show("Не удалось подключиться");

}

}

}

public partial class MainWindow : Window

{

public RolesViewModel ViewModel { get; set; }

public MainWindow()

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

label3.Content = UserAuth.isAdmin? "Администратор": UserAuth.isEmployee? "Сотрудник": UserAuth.isEnrollee? "Абитуриент" :"";

label2.Content = UserAuth.getUserAuth().User;

}

private void button\_Click(object sender, RoutedEventArgs e)

{

SQLAdmin sqlAdmin = new SQLAdmin();

sqlAdmin.Show();

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

CreateUser createUser = new CreateUser();

createUser.Show();

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

Person person = new Person();

person.Show();

}

private void button3\_Click(object sender, RoutedEventArgs e)

{

Enrollee enrollee = new Enrollee();

enrollee.Show();

}

private void button5\_Click(object sender, RoutedEventArgs e)

{

Formed formed = new Formed();

formed.Show();

}

private void button6\_Click(object sender, RoutedEventArgs e)

{

StatusForm statusForm = new StatusForm();

statusForm.Show();

}

private void button7\_Click(object sender, RoutedEventArgs e)

{

Paymont paymont = new Paymont();

paymont.Show();

}

private void button8\_Click(object sender, RoutedEventArgs e)

{

Discipline discipline = new Discipline();

discipline.Show();

}

private void button9\_Click(object sender, RoutedEventArgs e)

{

Vacancy vacancy = new Vacancy();

vacancy.Show();

}

private void button11\_Click(object sender, RoutedEventArgs e)

{

Speciality speciality = new Speciality();

speciality.Show();

}

private void button10\_Click(object sender, RoutedEventArgs e)

{

Exam exam = new Exam();

exam.Show();

}

private void button12\_Click(object sender, RoutedEventArgs e)

{

Discipline\_vacancy discipline\_Vacancy = new Discipline\_vacancy();

discipline\_Vacancy.Show();

}

private void button13\_Click(object sender, RoutedEventArgs e)

{

ExamResult examResult = new ExamResult();

examResult.Show();

}

private void button4\_Click(object sender, RoutedEventArgs e)

{

ChangePass changePass = new ChangePass();

changePass.Show();

}

private void button14\_Click(object sender, RoutedEventArgs e)

{

ExamResultPerson examResultPerson = new ExamResultPerson();

examResultPerson.Show();

}

private void button15\_Click(object sender, RoutedEventArgs e)

{

this.Content = new LoginPage();

UserAuth.getUserAuth().CloseConnectToDB();

}

}

public partial class Paymont : Window

{

isenrolleeDataSet dataTable;

paymontTableAdapter paymontDataAdapter;

public RolesViewModel ViewModel { get; set; }

public Paymont()

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

UserAuth con = UserAuth.getUserAuth();

paymontDataAdapter = new paymontTableAdapter();

paymontDataAdapter.Connection = con.con;

dataTable = new isenrolleeDataSet();

paymontDataAdapter.Fill(dataTable.paymont);

dataGrid.ItemsSource = dataTable.paymont.DefaultView;

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

/\* Сохранение \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

selectedItem[0] = textBox.Text;

selectedItem[1] = textBox1.Text;

}

paymontDataAdapter.Adapter.UpdateCommand = new NpgsqlCommandBuilder(paymontDataAdapter.Adapter).GetUpdateCommand();

paymontDataAdapter.Adapter.Update(dataTable.paymont);

}

private void dataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

/\* Заполнение полей по клику \*/

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

textBox.Text = selectedItem[0].ToString();

textBox1.Text = selectedItem[1].ToString();

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

/\* Создание \*/

paymontDataAdapter.Adapter.InsertCommand = new NpgsqlCommandBuilder(paymontDataAdapter.Adapter).GetInsertCommand();

paymontDataAdapter.InsertQuery1();

// обноляем записи в таблице datagrid

paymontDataAdapter.Fill(dataTable.paymont);

// выбираем последнюю запись, это будет та что создали

dataGrid.SelectedIndex = dataTable.paymont.Count - 1;

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

/\* Удаление \*/

int index = dataGrid.SelectedIndex;

var selectedItem = dataGrid.SelectedItem;

if (selectedItem != null)

{

dataTable.paymont.DefaultView.Delete(index);

}

paymontDataAdapter.Adapter.DeleteCommand = new NpgsqlCommandBuilder(paymontDataAdapter.Adapter).GetDeleteCommand();

paymontDataAdapter.Adapter.Update(dataTable.paymont);

}

}

public partial class Person : Window

{

personTableAdapter personDataAdapter;

isenrolleeDataSet dataTable;

UserAuth con;

public RolesViewModel ViewModel { get; set; }

public Person()

{

InitializeComponent();

ViewModel = new RolesViewModel();

this.DataContext = ViewModel;

con = UserAuth.getUserAuth();

try

{

personDataAdapter = new personTableAdapter();

personDataAdapter.Connection = con.con;

dataTable = new isenrolleeDataSet();

personDataAdapter.Fill(dataTable.person);

dataGrid.ItemsSource = dataTable.person.DefaultView;

}

catch (NpgsqlException ex)

{

MessageBox.Show(ex.ToString());

}

}

private void dataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

textBox.Text = selectedItem[0].ToString();

textBox1.Text = selectedItem[1].ToString();

textBox2.Text = selectedItem[2].ToString();

textBox3.Text = selectedItem[3].ToString();

textBox4.Text = selectedItem[4].ToString();

textBox5.Text = selectedItem[5].ToString();

datePicker.Text = selectedItem[6].ToString();

textBox6.Text = selectedItem[7].ToString();

textBox7.Text = selectedItem[8].ToString();

}

}

private void button\_Click(object sender, RoutedEventArgs e)

{

personDataAdapter.newPersonRow();

// обноляем записи в таблице datagrid

personDataAdapter.Fill(dataTable.person);

// выбираем последнюю запись, это будет та что создали

dataGrid.SelectedIndex = dataTable.person.Count - 1;

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

int index = dataGrid.SelectedIndex;

var selectedItem = dataGrid.SelectedItem;

if (selectedItem != null)

{

dataTable.person.DefaultView.Delete(index);

}

personDataAdapter.Adapter.DeleteCommand = new NpgsqlCommandBuilder(personDataAdapter.Adapter).GetDeleteCommand();

personDataAdapter.Update(dataTable.person);

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

selectedItem.BeginEdit();

//selectedItem[0] = textBox.Text;

selectedItem[1] = textBox1.Text;

selectedItem[2] = textBox2.Text;

selectedItem[3] = textBox3.Text;

selectedItem[4] = textBox4.Text;

selectedItem[5] = textBox5.Text;

selectedItem[6] = datePicker.SelectedDate;

selectedItem[7] = textBox6.Text;

selectedItem[8] = textBox7.Text;

selectedItem.EndEdit();

}

personDataAdapter.Adapter.UpdateCommand = new NpgsqlCommandBuilder(personDataAdapter.Adapter).GetUpdateCommand();

personDataAdapter.Update(dataTable.person);

}

private void button3\_Click(object sender, RoutedEventArgs e)

{

DataRowView selectedItem = dataGrid.SelectedItem as DataRowView;

if (selectedItem != null)

{

Enrollee enrollee = new Enrollee(Convert.ToInt32(selectedItem[0]));

enrollee.Show();

}

else

{

MessageBox.Show("Не выбран абитуриент!");

}

}

private void button4\_Click(object sender, RoutedEventArgs e)

{personDataAdapter.FillByFIO(dataTable.person,textBox8.Text,textBox8\_Copy.Text,textBox8\_Copy1.Text);

}

private void button5\_Click(object sender, RoutedEventArgs e)

{

textBox8.Text = "";

textBox8\_Copy.Text = "";

textBox8\_Copy1.Text = "";

personDataAdapter.Fill(dataTable.person);

}

}

public class RolesViewModel : INotifyPropertyChanged

{

private bool adminVisible = UserAuth.isAdmin;

private bool employeeVisible = UserAuth.isEmployee;

private bool enrolleeVisible = UserAuth.isEnrollee;

public bool AdminVisible

{

get

{

return adminVisible;

}

set

{

adminVisible = value;

NotifyPropertyChanged("AdminVisible");

}

}

public bool EmployeeVisible

{

get

{

return employeeVisible;

}

set

{

employeeVisible = value;

NotifyPropertyChanged("EmployeeVisible");

}

}

public bool EnrolleeVisible

{

get

{

return enrolleeVisible;

}

set

{

enrolleeVisible = value;

NotifyPropertyChanged("EnrolleeVisible");

}

}

private void NotifyPropertyChanged(string info)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(info));

}

}

public event PropertyChangedEventHandler PropertyChanged;

}

internal class UserAuth

{

static UserAuth context = null;

public NpgsqlConnection con = null;

string Host = "127.0.0.1";

public string User = "isenrollee";

string DBname = "isenrollee";

string Password = "isenrollee";

string Port = "5432";

UserRole Role = UserRole.None;

public int person\_id = 0;

public UserRole getRole()

{

return Role;

}

public static bool isAdmin => (getUserAuth().Role == UserRole.Admin);

public static bool isEmployee => ((getUserAuth().Role == UserRole.Employee) || (getUserAuth().Role == UserRole.Admin));

public static bool isEnrollee => (getUserAuth().Role == UserRole.Enrollee);

public static bool isNone=> (getUserAuth().Role == UserRole.None);

public static UserAuth getUserAuth()

{

if(context == null)

{

context = new UserAuth();

}

return context;

}

public UserAuth(string \_Host, string \_User, string \_DBname, string \_Password, string \_Port)

{

context = this;

Host = \_Host;

User = \_User;

DBname = \_DBname;

Password = \_Password;

Port = \_Port;

}

public UserAuth(string \_User, string \_Password)

{

context = this;

User = \_User;

Password = \_Password;

}

public UserAuth(){}

public bool ConnectToDB()

{

string connString =

String.Format(

"Server={0};Username={1};Database={2};Port={3};Password={4};SSLMode=Prefer",

Host,

User,

DBname,

Port,

Password);

con = new NpgsqlConnection(connString);

try

{

con.Open();

checkRoles();

}

catch

{

return false;

}

return true;

}

public bool changePass( string newPass)

{

var sqlPersonId = "ALTER ROLE current\_user WITH PASSWORD '"+ newPass + "';";

using (var cmd = new NpgsqlCommand(sqlPersonId, con))

{

try

{

cmd.ExecuteScalar();

}catch

{

return false;

}

}

return true;

}

private void checkRoles()

{

var sqlAdmin = "SELECT pg\_has\_role( current\_user,'admin', 'member')";

var sqlEmployee = "SELECT pg\_has\_role( current\_user,'employee','member')";

var sqlEnrollee = "SELECT pg\_has\_role( current\_user,'enrollee','member')";

var sqlPersonId= "SELECT person\_id FROM person WHERE (snils = current\_user)";

using (var cmd = new NpgsqlCommand(sqlPersonId, con))

{

var result = cmd.ExecuteScalar();

if (result != null)

{

person\_id = Convert.ToInt32(result);

}

}

using (var cmd = new NpgsqlCommand(sqlAdmin, con))

{

if ((bool)cmd.ExecuteScalar())

{

Role = UserRole.Admin;

return;

}

}

using (var cmd = new NpgsqlCommand(sqlEmployee, con))

{

if ((bool)cmd.ExecuteScalar())

{

Role = UserRole.Employee;

return;

}

}

using (var cmd = new NpgsqlCommand(sqlEnrollee, con))

{

if ((bool)cmd.ExecuteScalar())

{

Role = UserRole.Enrollee;

return;

}

}

Role = UserRole.None;

}

public void CloseConnectToDB()

{

con.Close();

}

~UserAuth()

{

CloseConnectToDB();

}

}

internal enum UserRole

{

Admin,

Employee,

Enrollee,

None

}

# Приложение Б Текст модульных тестов

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ИС\_Абитуриент;

using ИС\_Абитуриент.isenrolleeDataSetTableAdapters;

namespace TestProject2{

internal class Enrollee{

UserAuth auth;

isenrolleeDataSet isenrolleeDataSet;

[SetUp]

public void Setup(){

auth = new UserAuth("EnrolleeTest", "EnrolleeTest");

UserAuth.getUserAuth();

isenrolleeDataSet = new isenrolleeDataSet();

}

[Test]

public void Test1(){

Assert.IsTrue(auth.ConnectToDB(), "Нет подключения к бд");

}

[Test]

public void Test2(){

auth.ConnectToDB();

Assert.IsTrue(UserAuth.isEnrollee, "Роль пользователя не Абитуриент!");

}

[Test]

public void Test3(){

try{

auth.ConnectToDB();

paymontTableAdapter paymontDataAdapter;

paymontDataAdapter = new paymontTableAdapter();

paymontDataAdapter.Connection = UserAuth.getUserAuth().con;

paymontDataAdapter.Fill(isenrolleeDataSet.paymont); }

catch{

Assert.Fail();

}

Assert.Pass();

}

[Test]

public void Test4()

try{

auth.ConnectToDB();

disciplineTableAdapter DataAdapter;

DataAdapter = new disciplineTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.discipline);

}

catch{

Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test5()

try{

auth.ConnectToDB();

discipline\_vacancyTableAdapter DataAdapter;

DataAdapter = new discipline\_vacancyTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.discipline\_vacancy); }

catch{

Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test6(){

try{

auth.ConnectToDB();

enrolleeTableAdapter DataAdapter;

DataAdapter = new enrolleeTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.enrollee);

}

catch{

Assert.Fail();

Assert.Pass();

}

[Test]

public void Test7(){

try{

auth.ConnectToDB();

examTableAdapter DataAdapter;

DataAdapter = new examTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.exam);

catch{

Assert.Fail();

}

Assert.Pass();

}

[Test]

public void Test8(){

try{

auth.ConnectToDB();

enrolleeTableAdapter DataAdapter;

DataAdapter = new enrolleeTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.enrollee); }

catch{

Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test9()

{try{

auth.ConnectToDB();

formedTableAdapter DataAdapter;

DataAdapter = new formedTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.formed);

catch{

Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test10()

try{

auth.ConnectToDB();

personTableAdapter DataAdapter;

DataAdapter = new personTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.person);

catch{

Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test11(){

try{

auth.ConnectToDB();

specialityTableAdapter DataAdapter;

DataAdapter = new specialityTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.speciality);

catch{

Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test12(){

try

auth.ConnectToDB();

vacancyTableAdapter DataAdapter;

DataAdapter = new vacancyTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.vacancy); }

catch

{Assert.Fail();

Assert.Pass();

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ИС\_Абитуриент;

using ИС\_Абитуриент.isenrolleeDataSetTableAdapters;

namespace TestProject2{

internal class Employee{

UserAuth auth;

isenrolleeDataSet isenrolleeDataSet;

[SetUp]

public void Setup()

auth = new UserAuth("EmployeeTest", "EmployeeTest");

UserAuth.getUserAuth();

isenrolleeDataSet = new isenrolleeDataSet();

}

[Test]

public void Test1(){

Assert.IsTrue(auth.ConnectToDB(), "Нет подключения к бд");

}

[Test]

public void Test2()

{auth.ConnectToDB();

Assert.IsTrue(UserAuth.isEmployee, "Роль пользователя не Сотрудник!"); }

[Test]

public void Test3()

{try

{auth.ConnectToDB();

paymontTableAdapter paymontDataAdapter;

paymontDataAdapter = new paymontTableAdapter();

paymontDataAdapter.Connection = UserAuth.getUserAuth().con;

paymontDataAdapter.Fill(isenrolleeDataSet.paymont);

catch{

Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test4()

{try

{auth.ConnectToDB();

disciplineTableAdapter DataAdapter;

DataAdapter = new disciplineTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.discipline); }

catch

{Assert.Fail();

}

Assert.Pass();

}

[Test]

public void Test5()

{try

{auth.ConnectToDB();

discipline\_vacancyTableAdapter DataAdapter;

DataAdapter = new discipline\_vacancyTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.discipline\_vacancy); }

catch

{Assert.Fail();

Assert.Pass();

}

[Test]

public void Test6()

{try

{auth.ConnectToDB();

enrolleeTableAdapter DataAdapter;

DataAdapter = new enrolleeTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.enrollee);

catch

{Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test7()

{try

{auth.ConnectToDB();

examTableAdapter DataAdapter;

DataAdapter = new examTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.exam); }

catch

{Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test8()

{try

{auth.ConnectToDB();

enrolleeTableAdapter DataAdapter;

DataAdapter = new enrolleeTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.enrollee);

catch

{Assert.Fail();}

Assert.Pass();

}

[Test]

public void Test9()

{try

{auth.ConnectToDB();

formedTableAdapter DataAdapter;

DataAdapter = new formedTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.formed); }

catch

{Assert.Fail();

}Assert.Pass();

}

[Test]

public void Test10()

{try

{auth.ConnectToDB();

personTableAdapter DataAdapter;

DataAdapter = new personTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.person); }

catch

{Assert.Fail();}

Assert.Pass();}

[Test]

public void Test11()

{try

{auth.ConnectToDB();

specialityTableAdapter DataAdapter;

DataAdapter = new specialityTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.speciality); }

catch

{Assert.Fail();}

Assert.Pass();}

[Test]

public void Test12()

{try

{auth.ConnectToDB();

vacancyTableAdapter DataAdapter;

DataAdapter = new vacancyTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.Fill(isenrolleeDataSet.vacancy); }

catch

{Assert.Fail();}

Assert.Pass();

} }}

[Test]

public void Test13()

{try

{auth.ConnectToDB();

personTableAdapter DataAdapter;

DataAdapter = new personTableAdapter();

DataAdapter.Connection = UserAuth.getUserAuth().con;

DataAdapter.newPersonRow();

// обноляем записи в таблице datagrid

DataAdapter.Fill(isenrolleeDataSet.person); }

catch

{Assert.Pass();}

Assert.Fail("У пользователя есть права на запись в таблиу person");

}}}

using ИС\_Абитуриент;

using System.Runtime.InteropServices;

using Npgsql;

namespace TestProject2{

public class Tests{

[SetUp]

public void Setup() {}

[Test]

public void Test1(){ UserAuth auth = UserAuth.getUserAuth();

Assert.IsTrue(auth.ConnectToDB(), "Нет подключения к бд"); }}}

# Приложение В Антиплагиат

