Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Структуры и базы данных»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  магистр технических наук, ассистент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Сыс |
|  | \_\_\_.\_\_\_\_.2021 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«**БАЗА ДАННЫХ ДЛЯ ПОДДЕРЖКИ РАБОТЫ ПРИЕМНОЙ КОМИССИИ**»**

БГУИР КП 1-39 03 02 006 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 983871  КАФТАНЧИКОВА Анастасия Олеговна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2021  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2021

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc83885997)

[1 Анализ предметной области и ее формализация для проектирования базы данных 6](#_Toc83885998)

[1.1 Описание предметной области 6](#_Toc83885999)

[1.2 Анализ информационных потребностей пользователей и предварительное описание запросов 11](#_Toc83886000)

[1.3 Определение требований и ограничений к базе данных с точки зрения предметной области 11](#_Toc83886001)

[1.4 Постановка решаемой задачи 11](#_Toc83886002)

[2 Проектирование базы данных для основного вида деятельности рассматриваемой предметной области 13](#_Toc83886003)

[2.1 Разработка инфологической модели предметной области базы данных 13](#_Toc83886004)

[2.2 Выбор и обоснование используемых типов данных и ограничений (доменов) 17](#_Toc83886005)

[2.3 Проектирование запросов к базе данных 18](#_Toc83886006)

[2.4 Программная реализация и документирование базы данных 19](#_Toc83886007)

[3 Применение разработанной базы данных 28](#_Toc83886008)

[3.1 Руководство пользователя 28](#_Toc83886009)

[3.2 Администрирование базы данных 35](#_Toc83886010)

[3.3 Реализация клиентских запросов 36](#_Toc83886011)

[3.4 Обоснование и реализация механизма обеспечения безопасности и сохранности данных 37](#_Toc83886012)

[Заключение 38](#_Toc83886013)

[Список использованных источников 39](#_Toc83886014)

[Приложение А (обязательное) Скрипт генерации базы данных 40](#_Toc83886015)

[Приложение Б (обязательное) Листинг программного кода 46](#_Toc83886016)

[Приложение В (обязательное) Отчет о проверке на заимствования 54](#_Toc83886017)

[Приложение Г (обязательное) Ведомость курсового проекта 55](#_Toc83886018)

# Введение

При каждом университете существует центральная приёмная комиссия, которая начинает свою работу задолго до начала приёма документов, а заканчивает – через месяц после окончания вступительных испытаний.

Это подразделение вуза предназначено для организации набора студентов, проведения вступительных экзаменов и зачисления в состав обучаемых лиц, прошедших по конкурсу

Абитуриенты – это выпускники школ и других средних учебных заведений, решившие поступать в данный университет.

Они напрямую контактируют с приемной комиссией при обращении с документами: когда их сдают, когда получают экзаменационный лист, если подают апелляции, а также при процедурах зачисления или возврата документов.

До наступления вступительных экзаменов приемная комиссия осведомляет абитуриентов об их расписании, правилах проведения вступительных испытаний, правилах зачисления в данный ВУЗ, перечни направлений и специальностей в соответствии с аккредитацией, системой и критериями оценки результатов экзаменов, условиями подачи и рассмотрения апелляций.

Основные цели курсового проекта:

* оптимизация работы приемной комиссии в университете;
* создание базы данных для работы приемной комиссии;
* разработка приложения для работы приемной комиссии.

Основные задачи курсового проекта:

* реализовать возможность вести списки поступающих на различные факультеты и специальности;
* реализовать формирование групп для сдачи вступительных экзаменов;
* реализовать формирование графиков сдачи вступительных экзаменов;
* выполнять ввод результатов сдачи экзаменов;
* определять проходной балл;
* формировать списки поступивших.

Практическая значимость курсового проекта заключается в возможности использования результатов для оптимизации работы приемной комиссии, сокращении времени на обработку данных абитуриентов, исключение человеческих ошибок при зачислении.

Курсовой проект выполнен самостоятельно, проверен в системе «Атиплагиат». Процент оригинальности составляет 88,99%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведен в приложении В.

# 1 Анализ предметной области и ее формализация для проектирования базы данных

## 1.1 Описание предметной области

Приемная комиссия – это подразделение ВУЗа, осуществляющее приемную кампанию на будущий учебный год. Она организовывает набор студентов, знакомит абитуриентов с правилами и распорядком учебного учреждения, документацией по аккредитации направлений и специальностей, критериями поступления.

Также проводит консультации по вопросам ведения обучения и, самое главное, выносит решение о поступлении или не поступлении абитуриента в данное учебное заведение.

Председателем приёмной комиссии, как правило, является ректор ВУЗа. Он несет ответственность за выполнение установленных контрольных цифр приема, соблюдение законодательных актов и нормативных документов по формированию контингента обучаемых, определяет обязанности членов приемной комиссии, утверждает план ее работы и расписание вступительных испытаний.

Приказом ректора до 1 января назначается состав приёмной комиссии, куда входят:

* заместители председателя;
* ответственный секретарь и его заместители;
* заведующие кафедр;
* председатели предметных экзаменационных комиссий;
* сотрудники от профессорско-преподавательского состава университета.

Ответственный секретарь приемной комиссии может выполнять названные обязанности не более трех лет подряд. Для ведения и оформления документации из числа работников ВУЗа утверждаются технические секретари.

Для оказания помощи в работе приемных комиссий могут привлекаться участники студенческих отрядов.

Заместителем председателя приемной комиссии назначается проректор по учебной работе.

Срок полномочий приемной комиссии составляет один год.

Для приема вступительных экзаменов, своевременной подготовки необходимых материалов, объективности оценки способностей и склонностей, поступающих на первый курс приказом ректора вуза, создаются предметные экзаменационные комиссии и назначаются их председатели. Состав предметных экзаменационных комиссий формируется из числа наиболее опытных и квалифицированных научно-педагогических работников ВУЗа.

Приёмная кампания в ВУЗе – процесс, происходящий во времени. Его этапы:

* подготовительный этап;
* этап приёма документов у абитуриентов.
* этап приёма экзаменов;
* этап зачисления в университет.

Рассмотрим этапы подробнее.

На подготовительном этапе абитуриентам выдается информация о структуре ВУЗа и условиях приема в него.

Далее абитуриенты подают в приемную комиссию следующие документы:

* паспорт;
* оригиналы документов об образовании и приложения к ним;
* медицинскую справку о состоянии здоровья по форме, установленной Министерством здравоохранения;
* документы, подтверждающие право абитуриента на льготы [1].

Во время подачи документов абитуриент выбирает льготы, однако члены приемной комиссии должны подтвердить действительность поданной информации или скорректировать ее.

Рассмотрим возможные льготы абитуриентов.

Абитуриенты, которые могут поступать в вуз без вступительных испытаний:

* победители (дипломы I, II или III степени) международных олимпиад и республиканских олимпиад по первому профильному предмету;
* победители (Гран-при, дипломы I, II, III степени) международных и республиканских творческих конкурсов, и фестивалей, олимпиад по предметам культуры;
* участники Олимпийских, Паралимпийских и Дефлимпийских игр, лица, занявшие 1−3 места на Юношеских Олимпийских играх;
* абитуриенты, занявшие 1−3 места на чемпионатах, в финалах кубков, на первенствах мира и Европы, Европейских играх, Европейских юношеских олимпийских фестивалях;
* абитуриенты, занявшие 1−3 места на чемпионатах, первенствах мира и Европы среди инвалидов по зрению и/или с нарушением опорно-двигательного аппарата, на чемпионатах, первенствах мира и Европы среди инвалидов по слуху;
* абитуриенты, занявшие в год приёма или в году, предшествующем году приема, 1-е место на официальных чемпионатах Республики Беларусь;
* лица, награждённые в течение последних двух лет на момент получения общего среднего, среднего специального образования нагрудными знаками «Лаўрэат спецыяльнага фонду Прэзiдэнта Рэспублiкi Беларусь па сацыяльнай падтрымцы здольных навучэнцаў i студэнтаў» и «Лаўрэат спецыяльнага фонду Прэзiдэнта Рэспублiкi Беларусь па падтрымцы таленавiтай моладзi» за творческие достижения в сфере культуры;
* лица, имеющие аттестат об общем среднем образовании особого образца с награждением золотой или серебряной медалью, или имеющие диплом о среднем специальном образовании с отличием;
* выпускники учреждения образования «Минское суворовское военное училище», закончившие в год поступления данное учреждение с отметками 7 (семь) и выше баллов по всем предметам учебного плана либо закончившие специализированные по спорту классы и направленные в пределах плана распределения для дальнейшего обучения;
* выпускники государственного учреждения образования «Специализированный лицей при Университете гражданской защиты Министерства по чрезвычайным ситуациям Республики Беларусь», учреждения образования «Специализированный лицей Министерства внутренних дел Республики Беларусь» и кадетских училищ, закончившие в год поступления данные учреждения с отметками 7 (семь) и выше баллов по всем предметам учебного плана для дальнейшего обучения;
* лица, прошедшие обучение в профильных педагогических классах при наличии в документе об образовании отметок не ниже 7 (семи) баллов по всем учебным предметам и отметок не ниже 8 (восьми) баллов по профильным предметам при наличии рекомендации педагогического совета учреждения образования;
* победители (дипломы I, II, III степени) международных олимпиад и республиканской олимпиады по второму профильному предмету;
* победители (дипломы I, II, III степени) третьего (областного, Минского городского) этапа республиканской олимпиады по второму профильному предмету.

На места, оставшиеся после зачисления абитуриентов на основании пунктов выше, вне конкурса при наличии в документе об образовании отметок не ниже 6 (шести) баллов по предметам вступительных испытаний зачисляются:

* дети-сироты и дети, оставшиеся без попечения родителей, а также лица из числа детей-сирот и детей, оставшихся без попечения родителей;
* лица, занявшие в год приема или в году, предшествующем году приема, 2-е и 3-е места на официальных чемпионатах Республики Беларусь, 1-е место в официальных финалах кубков, на первенствах, Олимпийских днях молодежи Республики Беларусь по видам спорта, входящим в программу Олимпийских игр и включенным в реестр видов спорта Республики Беларусь;
* выпускники средних школ – училищ олимпийского резерва, получившие в год приема среднее специальное образование за счет средств бюджета;
* выпускники учреждения образования «Минское суворовское военное училище», государственного учреждения образования «Специализированный лицей при Университете гражданской защиты Министерства по чрезвычайным ситуациям Республики Беларусь», учреждения образования «Специализированный лицей Министерства внутренних дел Республики Беларусь» и кадетских училищ, закончившие в год поступления данные учреждения;
* лица, имеющие рекомендации воинских частей, лица рядового и младшего начальствующего состава органов внутренних дел, имеющие рекомендации органов внутренних дел;
* лица, имеющие рекомендации органов пограничной службы;
* лица, проходившие срочную военную службу в штатных подразделениях почетного караула Вооруженных Сил, органов пограничной службы, внутренних войск Министерства внутренних дел, имеющие рекомендацию должностного лица, осуществляющего общее руководство сводной ротой почетного караула при подготовке и проведении государственных торжественных мероприятий, и поступающие для получения первого высшего образования;

Преимущественное право на зачисление при равной общей сумме баллов в порядке перечисления имеют абитуриенты:

* получившие более высокий балл на вступительном испытании по первому предмету профильного испытания;
* получившие более высокий балл на вступительном испытании по второму предмету профильного испытания;
* имеющие более высокий балл в документе об образовании по первому предмету профильного испытания;
* имеющие более высокий балл в документе об образовании по второму предмету профильного испытания;
* являющиеся победителями республиканского молодежного конкурса «100 идей для Беларуси», при поступлении на специальности, соответствующие профилю их конкурсного проекта;
* являющиеся членами волонтерского движения «Доброе сердце» общественного объединения «Белорусский республиканский союз молодежи»;
* являющиеся членами молодежных отрядов охраны правопорядка, имеющими рекомендации Центрального комитета общественного объединения «Белорусский республиканский союз молодежи», при поступлении в УВО Министерства обороны и Министерства по чрезвычайным ситуациям;
* имеющие более высокий средний балл документа об образовании [2].

Подав документы, абитуриенты получают выписку с их регистрационным номером.

Этап приема экзаменов проходит следующим образом. Абитуриенты сдают два вступительных испытания по двум учебным предметам в соответствии с избранной специальностью в форме тестирования.

И на конечном этапе предметные экзаменационные комиссии обрабатывают результаты экзаменов и выносят решения о поступлении или не поступлении каждого абитуриента в данный ВУЗ, а также выдают абитуриентам эти результаты.

Таким образом для данной предметной области необходимы следующие таблицы:

Таблица «ABITURIENTS» содержит информацию об абитуриентах, подающих заявления в ВУЗ.

Таблица «GENDERS» является настроечной, в ней статичная информация о поле абитуриентов.

Таблица «FAMILY\_STATUS» является настроечной, в ней статичная информация о возможном семейном статусе.

Таблица «AB\_STATUS» является настроечной, в ней статичная информация о статусе абитуриента, например, документы поданы, забраны.

Таблица «PRIVILEGES» содержит информацию о всех льготах страны.

Таблица «DOCUMENT\_TYPES» содержит информацию о документах, удостоверяющих личность, которые возможны при подаче заявления в ВУЗ.

Таблица «SPECIALTIES» содержит информацию о специальностях в ВУЗе, на которые могут поступать абитуриенты.

Таблица «FACULTIES» содержит информацию о факультетах ВУЗа.

Таблица «PROFESSORS» содержит информацию о преподавателях в ВУЗе.

Таблица «EXAM\_LISTS» содержит информацию об экзаменах.

Таблица «SUBJECTS» содержит информацию о вступительных экзаменах.

Таблица «TIMETABLE» содержит информацию о расписании вступительных экзаменов.

Таблица «TIMETABLE\_TYPES» содержит информацию о типе мероприятия, запланированном в расписании.

Таблица «PRIORITY\_OF\_ SPECIALTIES» вспомогательная и служит для создания связи «многие-ко-многим» между абитуриентом и специальностями, уточняя приоритет.

Таблица «STUDENTS» содержит информацию о зачисленных студентах.

Таблица «FORMS\_STUDY» содержит информацию о форме получения образования.

Таблица «STATUS\_PRIVS» содержит информацию о типе льгот.

Таким образом, проектируемая база данных будет содержать всю необходимую информацию.

## 1.2 Анализ информационных потребностей пользователей и предварительное описание запросов

Пользователь БД – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

На каждом этапе развития базы данных (проектирование, реализация, эксплуатация, модернизация и развитие, полная реорганизация) с ней связаны разные категории пользователей.

Конечные пользователи – это основная категория пользователей, в интересах которых создается БД.

Администратор базы данных – это лицо или группа лиц, отвечающих за выработку требований к базе данных, ее проектирование, создание, эффективное использование и сопровождение.

Разработчики и администраторы приложений – это группа пользователей, которая функционирует во время проектирования, создания и реорганизации БД. Администраторы приложений координируют работу разработчиков при разработке конкретного приложения или группы приложений, объединенных в функциональную подсистему.

В разрабатываемой БД будет существовать следующие группы пользователей: администратор, абитуриент, преподаватель и член приемной комиссии. Для каждой группы пользователей будет существовать свои ограничения на использование данных, кроме администратора БД, который имеет полную свободу действий и может проводить с БД любые манипуляции.

Преподаватель, будет иметь ограничения на изменение и удаление личных данных, внесенными абитуриентами. Однако будет возможность проставления и изменения отметок за экзамены.

Члены приемной комиссии, будут иметь возможность только на изменение и удаление личных данных абитуриентов.

Абитуриент будет иметь ограничения на обновления, изменения и удаления данных.

## 1.3 Определение требований и ограничений к базе данных с точки зрения предметной области

Ограничения и требования к хранимым в базе данным следующие:

* номер паспорта абитуриента не должен повторяться;
* имя и фамилия на русском и латинице должны быть заполнены.

Также при подаче документов на форме приложения будет отмечены поля необходимые для заполнения в качестве списка, в котором абитуриент сможет выбрать не заполняя.

## 1.4 Постановка решаемой задачи

Целью курсового проектирования является построение базы данных для поддержки работы приемной комиссии.

Главная задача моделируемой системы – сохранение в базе данных всех необходимых сведений об абитуриентах, их результатов экзаменов, зачисленных в итоге студентов; осуществление поиска по заданному критерию; предоставление данных в удобном для пользователя виде.

Требования к целостности данных: данные в базе данных в любой момент времени должны быть правильными и непротиворечивыми. Требования к безопасности: доступ к системе должен быть осуществлён после аутентификации пользователя.

БД будет содержать:

* сведения об абитуриентах;
* сведения о преподавательском составе;
* сведения о специальностях ВУЗа;
* сведения о факультетах ВУЗа;
* сведения о льготах при поступлении;
* сведения об экзаменах;
* сведения о зачисленных абитуриентах.

Для достижения цели курсового проекта необходимо:

* реализовать способ подачи электронного заявления через приложение;
* реализовать электронный кабинет абитуриента, в котором будут отображаться личные данные, специальность и факультет на которые абитуриент поступает, а также расписание экзаменов и их результаты;
* реализовать возможность преподавателю проставление отметок за экзамены;
* реализовать процедуру проставления среднего балла за экзамены;
* реализовать процедуру расчёта итогового балла каждого абитуриента;
* реализовать процедуру зачисления абитуриентов по среднему баллу;
* реализовать процедуру расчета проходного балла по каждой специальности.

Кроме того, программа должна иметь дружественный и приятный интерфейс.

# 2 Проектирование базы данных для основного вида деятельности рассматриваемой предметной области

## 2.1 Разработка инфологической модели предметной области базы данных

Инфологическая модель описывает предметную область на содержательном уровне. Результатом этого анализа являются списки объектов предметной области, перечни свойств, или атрибутов, определение связей между объектами и описание структуры предметной области в виде диаграммы.

Определим связи данной предметной области на этапе разработки инфологической модели.

Связь между сущностями можно охарактеризовать степенью связи и классом принадлежности сущности к связи. Где степень связи показывает, сколько экземпляров одной сущности могут быть связано с каждым экземпляром другой сущности, и может иметь три значения:

* один к одному (1:1);
* один ко многим (1:М или М:1);
* многие ко многим (М:N).

Класс принадлежности сущности к связи может быть обязательным (каждый экземпляр сущности обязательно должен быть связан с другой сущностью) и необязательным (каждый экземпляр сущности не требует связи с экземпляром другой сущности) [4].

В данном курсовом проекте будут использованы следующие связи.

На рисунке 2.1 представлена связь таблиц «PRIVELEGIES» и «STATUS\_PRIVS».

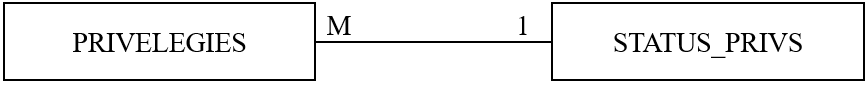


Рисунок 2.1 – Связь таблиц «PRIVELEGIES» и «STATUS\_PRIVS»

На рисунке 2.2 представлена связь таблиц «PRIVELEGIES» и «ABITURIENTS».

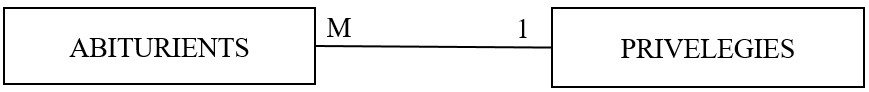


Рисунок 2.2 – Связь таблиц «PRIVELEGIES» и «ABITURIENTS»

На рисунке 2.3 представлена связь таблиц «FAMILY\_STATUS» и «ABITURIENTS».

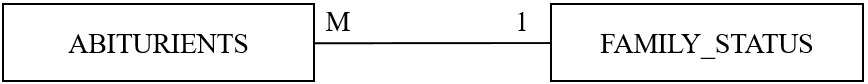


Рисунок 2.3 – Связь таблиц «FAMILY\_STATUS» и «ABITURIENTS»

На рисунке 2.4 представлена связь таблиц «DOCUMENT\_TYPES» и «ABITURIENTS».

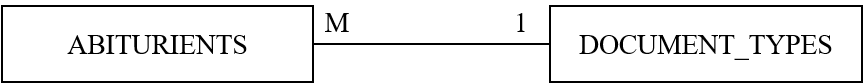


Рисунок 2.4 – Связь таблиц «DOCUMENT\_TYPES» и «ABITURIENTS»

На рисунке 2.5 представлена связь таблиц «AB\_STATUS» и «ABITURIENTS».

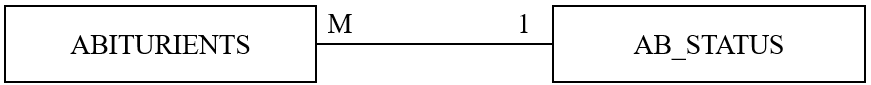


Рисунок 2.5 – Связь таблиц «AB\_STATUS» и «ABITURIENTS»

На рисунке 2.6 представлена связь таблиц «GENDERS» и «ABITURIENTS».

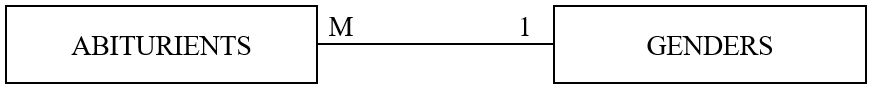


Рисунок 2.6 – Связь таблиц «GENDERS» и «ABITURIENTS»

На рисунке 2.7 представлена связь таблиц «PRIORITY\_OF\_SPECIALITIES» и «ABITURIENTS».

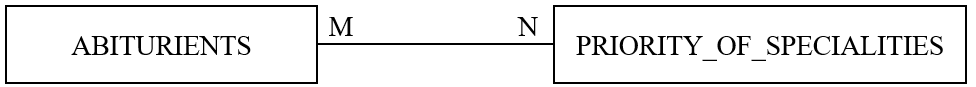


Рисунок 2.7 – Связь таблиц «PRIORITY\_OF\_SPECIALITIES» и «ABITURIENTS»

На рисунке 2.8 представлена связь таблиц «EXAM\_LISTS» и «ABITURIENTS».

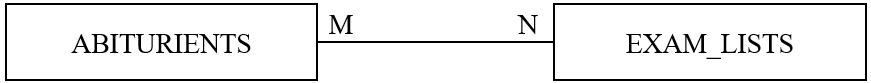


Рисунок 2.8 – Связь таблиц «EXAM\_LISTS» и «ABITURIENTS»

На рисунке 2.9 представлена связь таблиц «SPECIALITIES» и «PRIORITY\_OF\_SPECIALITIES».

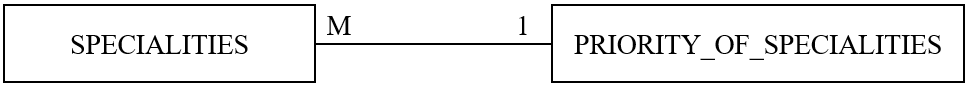


Рисунок 2.9 – Связь таблиц «SPECIALITIES» и «PRIORITY\_OF\_SPECIALITIES»

На рисунке 2.10 представлена связь таблиц «SPECIALITIES» и «FORMS\_STUDY».

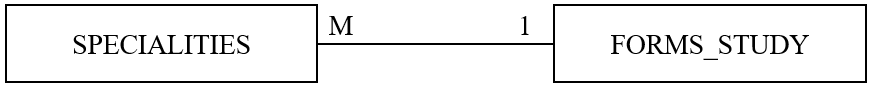


Рисунок 2.10 – Связь таблиц «SPECIALITIES» и «FORMS\_STUDY»

На рисунке 2.11 представлена связь таблиц «SPECIALITIES» и «FACULTIES».

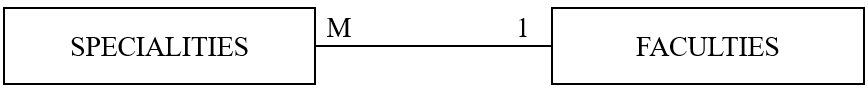


Рисунок 2.11 – Связь таблиц «SPECIALITIES» и «FACULTIES»

На рисунке 2.12 представлена связь таблиц «PROFESSORS» и «FACULTIES».

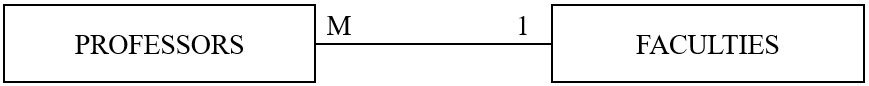


Рисунок 2.12 – Связь таблиц «PROFESSORS» и «FACULTIES»

На рисунке 2.13 представлена связь таблиц «PROFESSORS» и «EXAM\_LISTS».

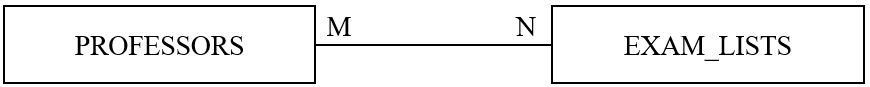


Рисунок 2.13 – Связь таблиц «PROFESSORS» и «EXAM\_LISTS»

На рисунке 2.14 представлена связь таблиц «SUBJECTS» и «EXAM\_LISTS».

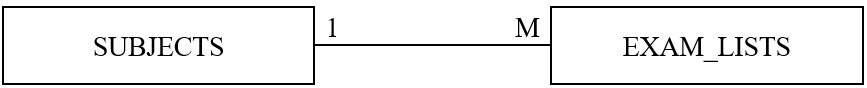


Рисунок 2.14 – Связь таблиц «SUBJECTS» и «EXAM\_LISTS»

На рисунке 2.15 представлена связь таблиц «TIMETABLE» и «SUBJECTS».

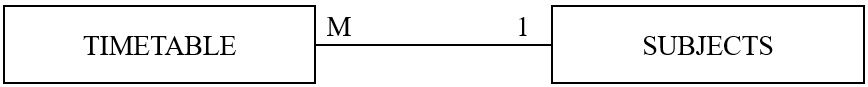


Рисунок 2.15 – Связь таблиц «TIMETABLE» и «SUBJECTS»

На рисунке 2.16 представлена связь таблиц «TIMETABLE» и «TIMETABLE\_TYPES».

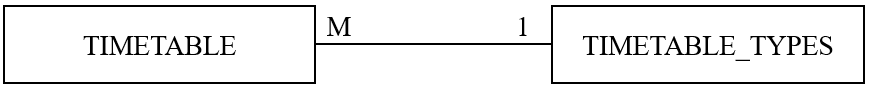


Рисунок 2.16 – Связь таблиц «TIMETABLE» и «TIMETABLE\_TYPES»

На рисунке 2.17 представлена связь таблиц «SPECIALITIES» и «TIMETABLE».

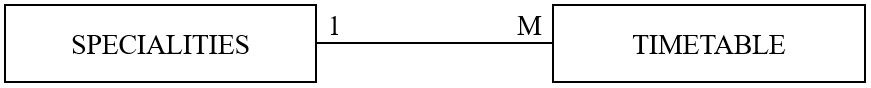


Рисунок 2.17 – Связь таблиц «SPECIALITIES» и «TIMETABLE»

На рисунке 2.18 представлена связь таблиц «STUDENTS» и «PRIVELEGIES».

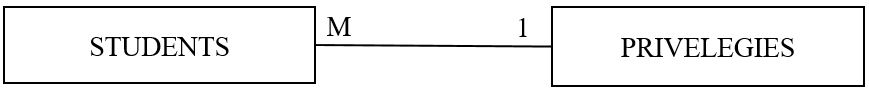


Рисунок 2.18 – Связь таблиц «STUDENTS» и «PRIVELEGIES»

На рисунке 2.19 представлена связь таблиц «STUDENTS» и «DOCUMENT\_TYPES».

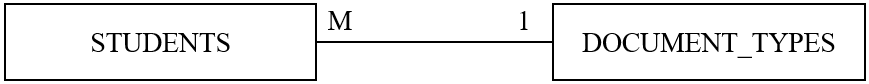


Рисунок 2.19 – Связь таблиц «STUDENTS» и «DOCUMENT\_TYPES»

На рисунке 2.20 представлена связь таблиц «STUDENTS» и «FAMILY\_STATUS».

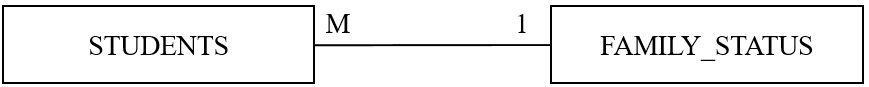


Рисунок 2.20 – Связь таблиц «STUDENTS» и «FAMILY\_STATUS»

На рисунке 2.21 представлена связь таблиц «STUDENTS» и «GENDERS».

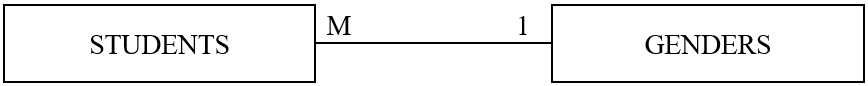


Рисунок 2.21 – Связь таблиц «STUDENTS» и «GENDERS»

На рисунке 2.22 представлена связь таблиц «STUDENTS» и «GENDERS».

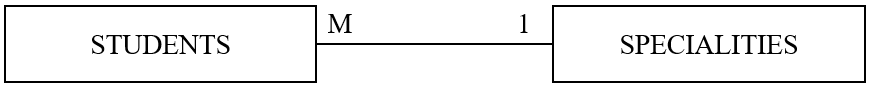


Рисунок 2.22 – Связь таблиц «STUDENTS» и «GENDERS»

Таким образом была разработана инфологическая модель предметной области базы данных, на основе которой построена структура БД.

## 2.2 Выбор и обоснование используемых типов данных и ограничений (доменов)

В данном курсовом проекте использованы такие типы данных как:

* int;
* varchar;
* date;
* numeric;
* datetime.

Тип INT хранит числа от –2 147 483 648 до 2 147 483 647. Занимает 4 байта. Используется для первичных и внешних ключей, а также для всех целочисленных данных.

Тип VARCHAR хранит строку. На каждый символ выделяется 1 байт. Можно указать конкретную длину для столбца - от 1 до 8 000 символов, например, VARCHAR (10). Если строка должна иметь больше 8000 символов, то задается размер MAX, а на хранение строки может выделяться до 2 Гб: VARCHAR(MAX). Не подходит для многих языков, так как хранит символы не в кодировке Unicode. В отличие от типа CHAR если в столбец с типом VARCHAR (10) будет сохранена строка в 5 символов, то в столбце будет сохранено именно пять символов. Используется для всех строковых данных.

Тип DATE хранит даты от 0001-01-01 (1 января 0001 года) до 9999-12-31 (31 декабря 9999 года). Занимает 3 байта. Используется для хранения дат рождения, выдачи и сроков действия документов, удостоверяющих личность.

Тип NUMERIC аналогичен типу DECIMAL. Тип DECIMAL хранит числа c фиксированной точностью. Занимает от 5 до 17 байт в зависимости от количества чисел после запятой. Данный тип может принимать два параметра precision и scale: DECIMAL (precision, scale). Параметр precision представляет максимальное количество цифр, которые может хранить число. Это значение должно находиться в диапазоне от 1 до 38. По умолчанию оно равно 18. Параметр scale представляет максимальное количество цифр, которые может содержать число после запятой. Это значение должно находиться в диапазоне от 0 до значения параметра precision. По умолчанию оно равно 0. Используется для хранения среднего балла за экзамены, по диплому и итоговый результат.

Тип DATETIME хранит даты и время от 01/01/1753 до 31/12/9999. Занимает 8 байт. Используется для составления расписания вступительных экзаменов [3].

## 2.3 Проектирование запросов к базе данных

Запросы упрощают просмотр, добавление, удаление или изменение данных в БД. Среди других целей использования запросов можно отметить:

* быстрый поиск определенных данных путем фильтрации с применением определенных критериев (условий);
* вычисление или сведение данных;
* автоматизированное управление данными, например, регулярный просмотр актуальных данных.

Для реализации возможности вести списки поступающих на различные факультеты и специальности будет выполняться команда INSERT. Однако до этого необходимо предоставить на выбор факультет и специальности. Выбор специальностей организован аналогично рисунку 2.23.

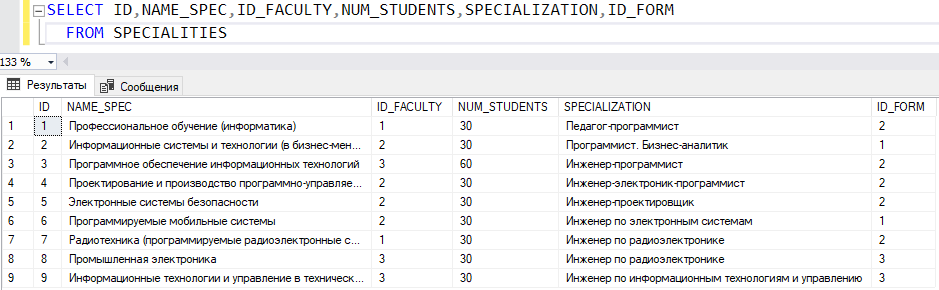


Рисунок 2.23 – Выбор специальностей

При этом можно будет сначала выбрать факультет, рисунок 2.24.

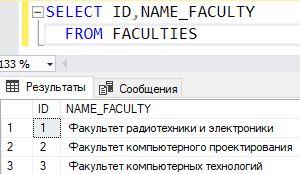


Рисунок 2.24 – Выбор факультета

Группы для сдачи вступительных экзаменов формируются по специальностям, в случае превышения абитуриентов в потоке, делится с помощью хранимой процедуры.

Для формирования графиков сдачи вступительных экзаменов реализуется следующий запрос, рисунок 2.25.

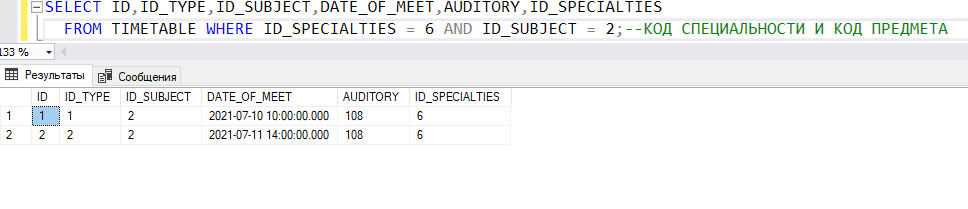


Рисунок 2.25 – Формирование расписания

Ввод результатов за экзамен будет осуществляться следующим запросом, рисунок 2.26.



Рисунок 2.26 – Ввод результатов за экзамен

Проходной балл определяется хранимой процедурой, суммируя средний балл по диплому и средний балл за экзамены.

Списки поступивших формируются следующим запросом, рисунок 2.27.

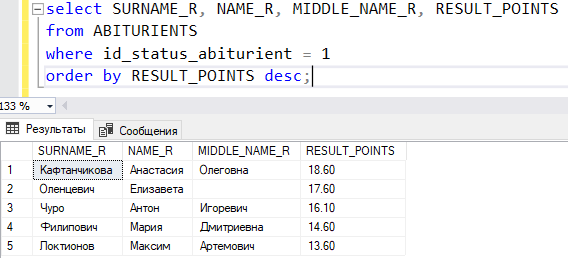


Рисунок 2.27 – Формирование списков поступивших

Таким образом реализованы запросы к БД.

## 2.4 Программная реализация и документирование базы данных

В реализуемой БД 17 сущностей. Рассмотрим описание каждой из них.

Сущность «GENDERS» содержит в себе информацию о гендере абитуриентов. Структура отображена в таблице 2.1.

Таблица 2.1 – Описание таблицы «GENDERS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| gender\_name | varchar(50) |  | Пол абитуриента | not null |

Сущность «FAMILY\_STATUS» содержит в себе информацию о семейном положении абитуриентов. Структура отображена в таблице 2.2.

Таблица 2.2 – Описание таблицы «FAMILY\_STATUS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| status\_name | varchar(255) |  | Семейное положение абитуриента | not null |

Сущность «DOCUMENT\_TYPES» содержит в себе информацию о типе документа, удостоверяющего личность абитуриента. Структура отображена в таблице 2.3.

Таблица 2.3 – Описание таблицы «DOCUMENT\_TYPES»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| doc\_name | varchar(255) |  | Документ, удостоверяющий личность | not null |

Сущность «PRIVELEGIES» содержит в себе информацию о льготах абитуриентов. Структура отображена в таблице 2.4.

Таблица 2.4 – Описание таблицы «PRIVELEGIES»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| priv\_name | varchar(1000) |  | Льготы | not null |
| status\_priv\_id | int | foreign key (status\_priv\_id) references status\_privs (id) | Внешний ключ к таблице «STATUS\_PRIVS» | not null |
| status\_active | int |  | Статус льгот | not null,  (1 – активный,  0 – не активный) |

Сущность «STATUS\_PRIVS» содержит в себе информацию о типе льгот. Структура отображена в таблице 2.5.

Таблица 2.5 – Описание таблицы «STATUS\_PRIVS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| status\_privs\_name | varchar(255) |  | Тип льгот | not null |

Сущность «AB\_STATUS» содержит в себе информацию о статусе абитуриента в текущий момент. Структура отображена в таблице 2.6.

Таблица 2.6 – Описание таблицы «AB\_STATUS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| abst\_name | varchar(255) |  | Статус абитуриента | not null |

Сущность «FACULTIES» содержит в себе информацию о факультетах университета. Структура отображена в таблице 2.7.

Таблица 2.7 – Описание таблицы «FACULTIES»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| name\_faculty | varchar(50) |  | Название факультета | not null |
| corpus | int |  | Номер корпуса |  |

Сущность «PROFESSORS» содержит в себе информацию о факультетах университета. Структура отображена в таблице 2.8.

Таблица 2.8 – Описание таблицы «PROFESSORS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| *1* | *2* | *3* | *4* | *5* |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| name\_prof | varchar(50) |  | ФИО преподавателя | not null |

Продолжение таблицы 2.8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1* | *2* | *3* | *4* | *5* |
| id\_faculty | int | foreign key (id\_faculty) references faculties (id) | Внешний ключ к таблице «FACULTIES» | not null |
| login | varchar(50) |  | Логин преподавателя |  |

Сущность «TIMETABLE\_TYPES» содержит в себе информацию о типе мероприятий приемной комиссии. Структура отображена в таблице 2.9.

Таблица 2.9 – Описание таблицы «TIMETABLE\_TYPES»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| name\_type | varchar(50) |  | Тип мероприятия | not null |

Сущность «SUBJECTS» содержит в себе информацию о типе мероприятий приемной комиссии. Структура отображена в таблице 2.10.

Таблица 2.10 – Описание таблицы «SUBJECTS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| name\_subject | varchar(100) |  | Название предмета | not null |

Сущность «SPECIALITIES» содержит в себе информацию о типе мероприятий приемной комиссии. Структура отображена в таблице 2.11.

Таблица 2.11 – Описание таблицы «SPECIALITIES»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| *1* | *2* | *3* | *4* | *5* |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| name\_spec | varchar(100) |  | Название специальности | not null |
| id\_faculty | int | foreign key (id\_faculty) references faculties (id) | Внешний ключ к таблице «FACULTIES» | not null |

Продолжение таблицы 2.11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1* | *2* | *3* | *4* | *5* |
| num\_students | int |  | Количество мест для абитуриентов | not null |
| specialization | varchar(100) |  | Специализация | not null |
| id\_form | int | foreign key (id\_form) references forms\_study (id) | Форма обучения в университете | not null |

Сущность «FORMS\_STUDY» содержит в себе информацию о форме обучения в университете. Структура отображена в таблице 2.12.

Таблица 2.12 – Описание таблицы «FORMS\_STUDY»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| name\_form | varchar(100) |  | Форма обучения | not null |

Сущность «ABITURIENTS» содержит в себе информацию о типе мероприятий приемной комиссии. Структура отображена в таблице 2.13.

Таблица 2.13 – Описание таблицы «ABITURIENTS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| *1* | *2* | *3* | *4* | *5* |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| surname\_r | varchar  (50) |  | Фамилия на русском | not null |
| name\_r | varchar  (50) |  | Имя на русском | not null |
| middle\_name\_r | varchar  (50) |  | Отчество на русском |  |
| surname\_l | varchar  (50) |  | Фамилия на латинице | not null |
| name\_l | varchar  (50) |  | Имя на латинице | not null |
| id\_gender | int | foreign key (id\_gender) references genders (id) | Внешний ключ к таблице «GENDERS» | not null |
| date\_of\_birth | date |  | Дата рождения | not null |

Продолжение таблицы 2.13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1* | *2* | *3* | *4* | *5* |
| id\_family\_status | int | foreign key (id\_family\_status) references family\_status (id) | Внешний ключ к таблице «FAMILY\_STATUS» | not null |
| id\_document\_type | int | foreign key (id\_document\_type) references document\_types (id) | Внешний ключ к таблице «DOCUMENT\_TYPES» | not null |
| number\_id | int |  | Идентификационный номер документа, удостоверяющего личность | not null |
| serial | varchar  (2) |  | Серия документа, удостоверяющего личность | not null |
| ser\_number | varchar  (7) |  | Номер документа, удостоверяющего личность | not null |
| issue\_author | varchar  (50) |  | Орган, выдавший документ, удостоверяющий личность | not null |
| date\_form\_issue | date |  | Дата выдачи документа, удостоверяющего личность | not null |
| date\_for\_issue | date |  | Дата окончания действия документа, удостоверяющего личность | not null |
| aver\_exam | numeric  (4,2) |  | Средний балл за экзамены |  |
| id\_privileges | int | foreign key (id\_privileges) references privelegies (id) | Внешний ключ к таблице «PRIVELEGIES» | not null |
| aver\_diplom | numeric  (4,2) |  | Средний балл по диплому | not null |
| result\_points | numeric  (4,2) |  | Итоговый результат |  |
| id\_status\_abiturient | int | foreign key (id\_status\_abiturient) references ab\_status (id) | Внешний ключ к таблице «AB\_STATUS» | not null |
| group\_exam | int |  | Группа на экзамен |  |
| login\_abi | varchar  (50) |  | Логин абитуриента |  |

Сущность «EXAM\_LISTS» содержит в себе информацию о вступительных экзаменах. Структура отображена в таблице 2.14.

Таблица 2.14 – Описание таблицы «EXAM\_LISTS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| id\_subject | int | foreign key (id\_subject) references subjects (id) | Внешний ключ к таблице «SUBJECTS» | not null |
| id\_abiturient | int | foreign key (id\_abiturient) references abiturients (id) | Внешний ключ к таблице «ABITURIENTS» | not null |
| id\_professor | int | foreign key (id\_professor) references professors (id) | Внешний ключ к таблице «PROFESSORS» | not null |
| exam\_grade | int |  | Отметка за экзамен | not null |

Сущность «PRIORITY\_OF\_SPECIALITIES» содержит в себе информацию о приоритете выбора специальности. Структура отображена в таблице 2.15.

Таблица 2.15 – Описание таблицы «PRIORITY\_OF\_SPECIALITIES»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| id\_abiturient | int | foreign key (id\_abiturient) references abiturients (id) | Внешний ключ к таблице «ABITURIENTS» | not null |
| id\_specialties | int | foreign key (id\_specialties) references specialities (id) | Внешний ключ к таблице «SPECIALITIES» | not null |
| priority | int |  | Приоритет выбора | not null |

Сущность «TIMETABLE» содержит в себе информацию о расписании вступительных экзаменов. Структура отображена в таблице 2.16.

Таблица 2.16 – Описание таблицы «TIMETABLE»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| id\_type | int | foreign key (id\_type) references (id) | Внешний ключ к таблице «TIMETABLE\_TYPES» | not null |
| id\_subject | int | foreign key (id\_subject) references subjects (id) | Внешний ключ к таблице «SUBJECTS» | not null |
| date\_of\_meet | datetime |  | Дата мероприятия | not null |
| auditory | varchar(10) |  | Аудитория | not null |
| id\_specialties | int | foreign key (id\_specialties) references specialities (id) | Внешний ключ к таблице «SPECIALITIES» | not null |

Сущность «STUDENTS» содержит в себе информацию о зачисленных абитуриентах. Структура отображена в таблице 2.17.

Таблица 2.17 – Описание таблицы «STUDENTS»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип данных | Ключ | Описание | Правила и ограничения |
| *1* | *2* | *3* | *4* | *5* |
| id | int | identity primary key | Уникальный идентификатор таблицы | not null |
| surname\_r | varchar  (50) |  | Фамилия на русском | not null |
| name\_r | varchar  (50) |  | Имя на русском | not null |
| middle\_name\_r | varchar  (50) |  | Отчество на русском |  |
| surname\_l | varchar  (50) |  | Фамилия на латинице | not null |
| name\_l | varchar  (50) |  | Имя на латинице | not null |
| id\_gender | int | foreign key (id\_gender) references genders (id) | Внешний ключ к таблице «GENDERS» | not null |
| date\_of\_birth | date |  | Дата рождения | not null |

Продолжение таблицы 2.17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1* | *2* | *3* | *4* | *5* |
| id\_family\_status | int | foreign key (id\_family\_status) references family\_status (id) | Внешний ключ к таблице «FAMILY\_STATUS» | not null |
| id\_document\_type | int | foreign key (id\_document\_type) references document\_types (id) | Внешний ключ к таблице «DOCUMENT\_TYPES» | not null |
| number\_id | int |  | Идентификационный номер документа, удостоверяющего личность | not null |
| serial | varchar  (2) |  | Серия документа, удостоверяющего личность | not null |
| ser\_number | varchar  (7) |  | Номер документа, удостоверяющего личность | not null |
| issue\_author | varchar  (50) |  | Орган, выдавший документ, удостоверяющий личность | not null |
| date\_form\_issue | date |  | Дата выдачи документа, удостоверяющего личность | not null |
| date\_for\_issue | date |  | Дата окончания действия документа, удостоверяющего личность | not null |
| num\_group |  |  | Номер группы | not null |
| id\_specialties | int | foreign key (id\_specialties) references specialities (id) | Внешний ключ к таблице «SPECIALITIES» | not null |
| id\_privileges | int | foreign key (id\_privileges) references privelegies (id) | Внешний ключ к таблице «PRIVELEGIES» | not null |

Таким образом была реализована и документирована БД.

# 3 Применение разработанной базы данных

## 3.1 Руководство пользователя

**3.1.1 Область применения**

Разработанное программное средство применяются для поддержки работы приемной комиссии.

**3.1.2 Краткое описание возможностей**

Программное средство позволяет:

* абитуриенту подавать документы на различные факультеты и специальности;
* формировать группы для вступительных экзаменов;
* формировать графики вступительных экзаменов;
* вводить результаты сдачи экзаменов;
* формировать списки поступивших.

**3.1.3 Уровень подготовки пользователя**

Пользователь программного средства должен иметь опыт работы с ОС MS Windows (7/8/10), навык работы с консольным ПО, а также обладать следующими знаниями:

* знать соответствующую предметную область;
* знать основы многомерного анализа;
* понимать многомерную модель соответствующей предметной области;
* знать и иметь навыки работы с структурными приложениями.

Квалификация пользователя должна позволять осуществлять анализ данных.

**3.1.4 Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю**

Указ Президента Республики Беларусь от 20.04.2021 №155 «Об организации вступительной кампании в учреждениях высшего и среднего специального образования в 2021 году», правила приема лиц для получения высшего образования I ступени.

**3.1.5 Состав и содержание дистрибутивного носителя данных**

Для работы с разработанным программным средством необходимо следующее программное обеспечение: SQL Server 2019 Express для проверки и отслеживания действий программного средства.

**3.1.6 Порядок загрузки данных и программ**

Перед началом работы с программным средством на рабочем месте пользователя необходимо скачать приложение подходящее ОС пользователя.

**3.1.7 Порядок проверки работоспособности**

Для проверки доступности разработанного программного средства с рабочего места пользователя необходимо дважды нажать левой кнопкой мыши на приложение или с помощью правой кнопки мыши, нажав на нее и выбрав Открыть.

В случае если приложение не запускается, то следует обратиться в службу поддержки.

**3.1.8 Описание операций технологического процесса обработки данных, необходимых для выполнения задач**

Задача: «Абитуриенту подать документы на различные факультеты и специальности»

Условия, при соблюдении которых возможно выполнение операции: разработанное программное средство функционирует в штатном режиме.

Подготовительные действия: на компьютере пользователя необходимо выполнить дополнительные настройки, приведенные в п. 3.1.5-3.1.7 настоящего документа.

Основные действия в требуемой последовательности:

* на иконке программного средства рабочего стола произвести двойной щелчок левой кнопкой мышки;
* в открывшемся окне выбрать «Подать заявление», рисунок 3.1;
* в открывшемся окне заполнить все поля со «\*», рисунок 3.2;
* после этого нажать кнопку «Подать», в случае правильного заполнения откроется сообщение с логином и паролем абитуриента, иначе сообщение об ошибке.

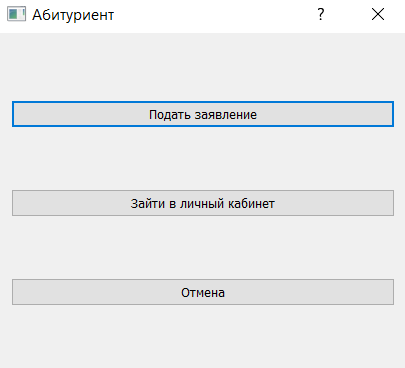


Рисунок 3.1 – Выбор на главном экране



Рисунок 3.2 – Заполнение заявления

Заключительные действия: не требуются.

Ресурсы, расходуемые на операцию: 5-10 секунд.

Задачи: «Формировать группы для вступительных экзаменов» и «Формировать графики вступительных экзаменов».

Условия, при соблюдении которых возможно выполнение операции: разработанное программное средство функционирует в штатном режиме.

Подготовительные действия: на компьютере пользователя необходимо выполнить дополнительные настройки, приведенные в п. 3.1.5-3.1.7 настоящего документа.

Основные действия в требуемой последовательности:

* на иконке программного средства рабочего стола произвести двойной щелчок левой кнопкой мышки;
* в открывшемся окне выбрать «Зайти в личный кабинет», рисунок 3.3;
* в открывшемся окне выбрать «Абитуриент», рисунок 3.4;
* в открывшемся окне ввести логин и пароль, выданные при подаче заявления, рисунок 3.5;
* после этого откроется личный кабинет абитуриента, рисунок 3.6, где будет сформировано расписание и номер группы на экзамены, а также после выставления отметок за экзамен, они будут отображаться там же.

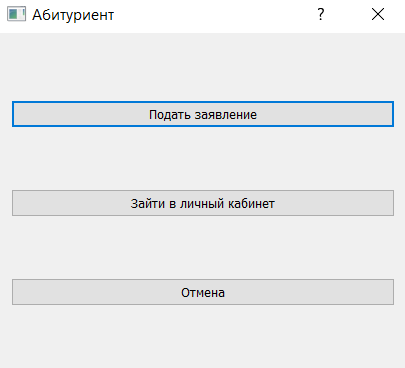


Рисунок 3.3 – Выбор на главном экране

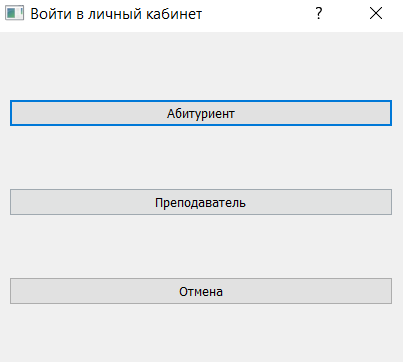


Рисунок 3.4 – Выбор на экране

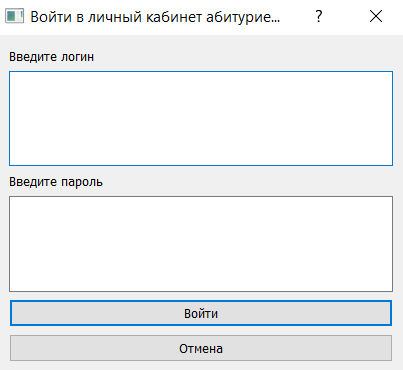


Рисунок 3.5 – Аутентификация пользователя



Рисунок 3.6 – Личный кабинет абитуриента

Заключительные действия: не требуются.

Ресурсы, расходуемые на операцию: 5-10 секунд.

Задача: «Вводить результаты сдачи экзаменов».

Условия, при соблюдении которых возможно выполнение операции: разработанное программное средство функционирует в штатном режиме.

Подготовительные действия: на компьютере пользователя необходимо выполнить дополнительные настройки, приведенные в п. 3.1.5-3.1.7 настоящего документа.

Основные действия в требуемой последовательности:

* на иконке программного средства рабочего стола произвести двойной щелчок левой кнопкой мышки;
* в открывшемся окне выбрать «Зайти в личный кабинет», рисунок 3.7;
* в открывшемся окне выбрать «Преподаватель», рисунок 3.8;
* в открывшемся окне ввести логин, пароль и код проверяемой дисциплины, выданные при подаче заявления, рисунок 3.9;
* после этого для каждого абитуриента будет открываться сообщение с его ФИО и идентификационным номером и полем для ввода отметок, рисунок 3.10, а после этого откроется личный кабинет преподавателя, рисунок 3.11, с введенными отметками.

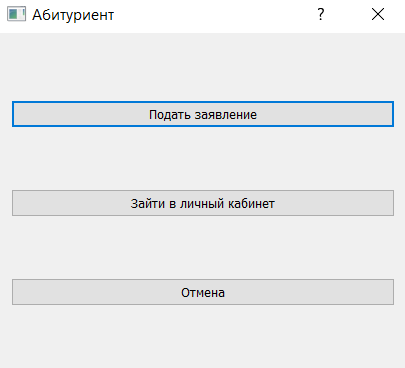


Рисунок 3.7 – Выбор на главном экране

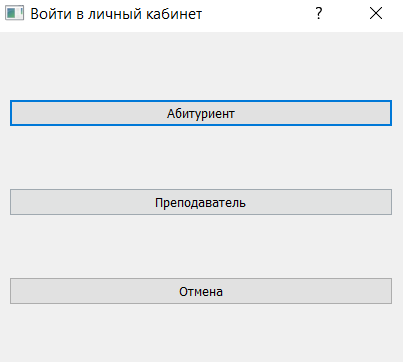


Рисунок 3.8 – Выбор на экране

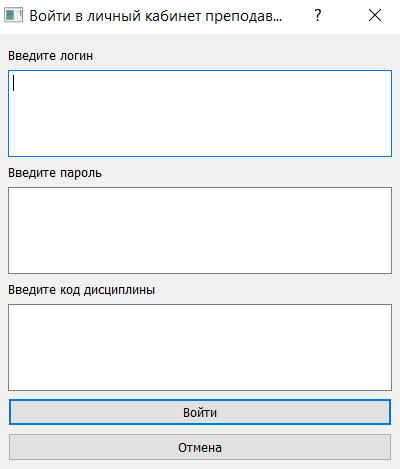


Рисунок 3.9 – Аутентификация пользователя

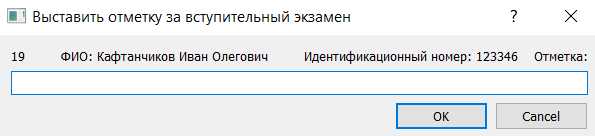


Рисунок 3.10 – Сообщение ввода отметок

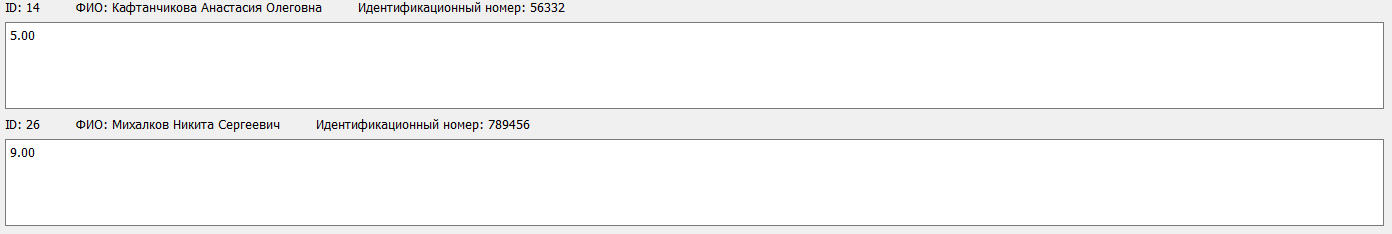


Рисунок 3.11 – Личный кабинет преподавателя

Заключительные действия: не требуются.

Ресурсы, расходуемые на операцию: 5-10 секунд.

Задачи: «Формировать списки поступивших».

Условия, при соблюдении которых возможно выполнение операции: разработанное программное средство функционирует в штатном режиме.

Подготовительные действия: на компьютере пользователя необходимо выполнить дополнительные настройки, приведенные в п. 3.1.5-3.1.7 настоящего документа.

Основные действия в требуемой последовательности:

* на иконке программного средства рабочего стола произвести двойной щелчок левой кнопкой мышки;
* в открывшемся окне выбрать «Зайти в личный кабинет», рисунок 3.12;
* в открывшемся окне выбрать «Абитуриент», рисунок 3.13;
* в открывшемся окне ввести логин и пароль, выданные при подаче заявления, рисунок 3.14;
* после этого откроется личный кабинет абитуриента, рисунок 3.15, будет отображен статус «Зачислен на курс».

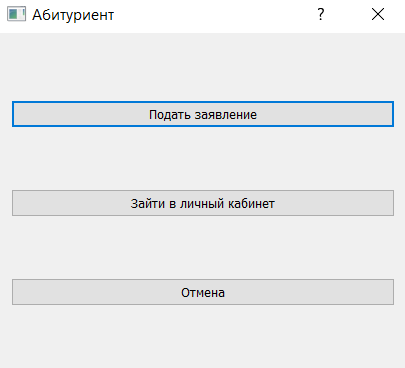


Рисунок 3.12 – Выбор на главном экране

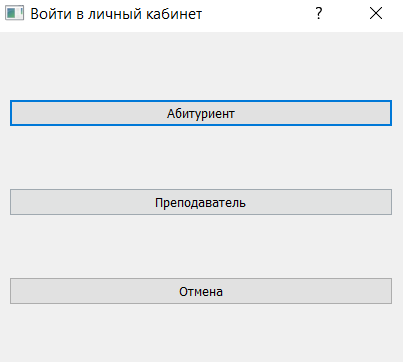


Рисунок 3.13 – Выбор на экране

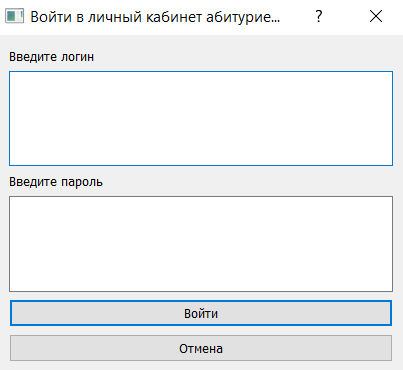


Рисунок 3.14 – Аутентификация пользователя

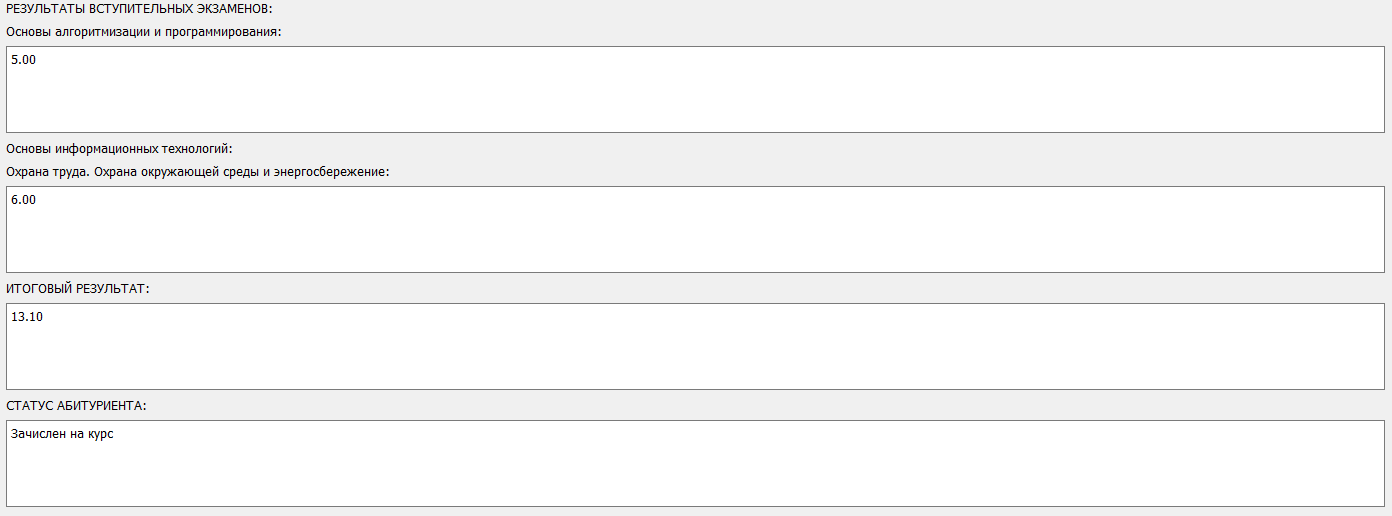


Рисунок 3.15 – Личный кабинет абитуриента

Заключительные действия: не требуются.

Ресурсы, расходуемые на операцию: 5-10 секунд.

**3.1.9 Аварийные ситуации**

В случае возникновения ошибок при работе программного средства, необходимо обращаться к сотруднику подразделения технической поддержки либо к ответственному Администратору.

**3.1.10 Рекомендации по освоению**

Рекомендуемая литература: правила проведения вступительных экзаменов.

В качестве контрольного примера рекомендуется выполнить операции задачи, описанные в п. 3.1.4 настоящего документа.

## 3.2 Администрирование базы данных

Администратор базы данных отвечает за целостность информационных ресурсов компании. На нем лежит ответственность по созданию, обновлению и сохранности связанных между собой резервных копий файлов, исходя из задач предприятия. Этот человек должен в мельчайших подробностях знать существующие механизмы восстановления программного обеспечения базы данных.

Возможны ситуации, при которых администратору базы данных потребуется на основе логических прикладных моделей создавать элементы физической схемы, а также поддерживать связь пользователей с системой и обеспечивать соответствующий уровень информационной безопасности, следя за тем, чтобы доступ к данным имели только те люди, которые в нем нуждаются.

Администратор базы данных должен уметь определять узкие места системы, ограничивающие ее производительность, настраивать SQL и программное обеспечение системы управления базами данных и обладать знаниями, необходимыми для решения вопросов оптимизации быстродействия базы данных [4].

Для ограничения доступа к данным, реализованы 2 роли: ABITURIENTS и PROFESSORS.

Для каждого пользователя создается отдельный пользователь в БД, которые добавляются к ролям.

Роли ABITURIENTS даны следующие GRANT-ы, рисунок 3.16.

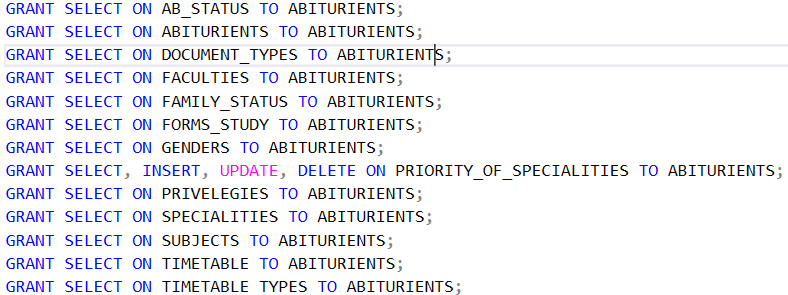


Рисунок 3.16 – GRANT-ы ABITURIENTS

Роли PROFESSORS даны следующие GRANT-ы, рисунок 3.17.



Рисунок 3.17 – GRANT-ы PROFESSORS

Таким образом происходит администрирование БД.

## 3.3 Реализация клиентских запросов

В данном курсовом проекте реализованы следующие виды SQL-запросов:

* DDL (Data Definition Language) – язык определения данных. Задачей DDL запросов является создание БД и описание ее структуры. Запросами такого вида устанавливаются правила того, в каком виде различные данные будут размещаться в БД;
* DML (Data Manipulation Language) – язык манипулирования данными. В число запросов этого типа входят различные команды, используя которые непосредственно производятся некоторые манипуляции с данными. DML-запросы нужны для добавления изменений в уже внесенные данные, для получения данных из БД, для их сохранения, для обновления различных записей и для их удаления из БД. В число элементов DML-обращений входит основная часть SQL операторов;
* DCL (Data Control Language) – язык управления данными. Включает в себя запросы и команды, касающиеся разрешений, прав и других настроек СУБД;
* TCL (Transaction Control Language) – язык управления транзакциями. Конструкции такого типа применяют чтобы управлять изменениями, которые производятся с использованием DML запросов. Конструкции TCL позволяют нам производить объединение DML запросов в наборы транзакций, рисунок 3.18 [5].

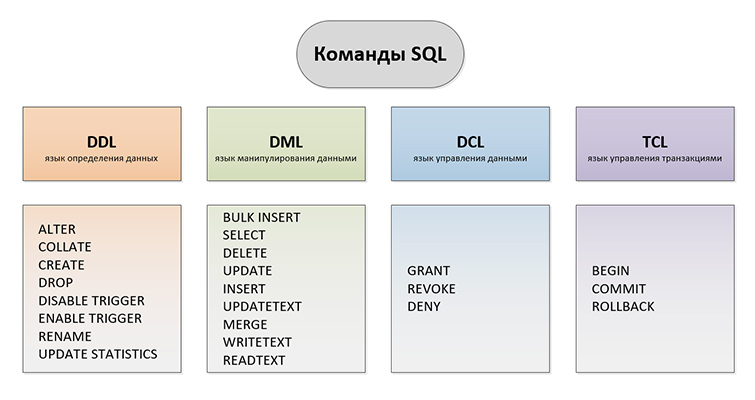


Рисунок 3.18 – Виды SQL-запросов

Таким образом были реализованы SQL-запросы.

## 3.4 Обоснование и реализация механизма обеспечения безопасности и сохранности данных

В современных условиях любая деятельность сопряжена с оперированием большими объемами информации, которое производится широким кругом лиц. Защита данных от несанкционированного доступа является одной из приоритетных задач при проектировании любой информационной системы. Следствием возросшего в последнее время значения информации стали высокие требования к конфиденциальности данных. Системы управления базами данных, в особенности реляционные систему управления базами данных, стали доминирующим инструментом в этой области. Обеспечение информационной безопасности системы управления базами данных приобретает решающее значение при выборе конкретного средства обеспечения необходимого уровня безопасности организации в целом.

Для систем управления базами данных важны три основных аспекта информационной безопасности – конфиденциальность, целостность и доступность.

Политика безопасности определяется администратором данных. Однако решения защиты данных не должны быть ограничены только рамками системы управления базами данных. Абсолютная защита данных практически не реализуема, поэтому обычно довольствуются относительной защитой информации – гарантированно защищают ее на тот период времени, пока несанкционированный доступ к ней влечет какие-либо последствия. Разграничение доступа к данным также описывается в базе данных посредством ограничений, и информация об этом хранится в ее системном каталоге. Иногда дополнительная информация может быть запрошена из операционных систем, в окружении которых работают сервер баз данных и клиент, обращающийся к серверу баз данных.

# Заключение

В данном курсовом проекте была создана база данных для поддержки работы приемной комиссии, а также клиентское приложение, которое позволяет:

* абитуриенту подавать документы на различные факультеты и специальности;
* формировать группы для вступительных экзаменов;
* формировать графики вступительных экзаменов;
* вводить результаты сдачи экзаменов;
* формировать списки поступивших.

Пользователь программного средства должен иметь опыт работы с ОС MS Windows (7/8/10), навык работы с консольным ПО, а также обладать соответствующими знаниями предметной области. Квалификация пользователя должна позволять осуществлять анализ данных.

Рекомендуемая литература: правила проведения вступительных экзаменов. В качестве контрольного примера рекомендуется выполнить операции задачи, описанные в п. 3.1.

В итоге выполнения курсового проекта стала ясна вся значимость и необходимость баз данных в современном мире.

Практически доказано, что технология использования баз данных для хранения, структуризации и анализа информации позволяет существенно сохранять человеческий ресурс на обработку данных об объектах реального мира.

В конечном счете, после выполнения курсового проекта были получены теоретические знания относительно проектирования и реализации баз данных, изучены принципы работы с языком SQL.

# Список использованных источников

[1] Положение о приемной комиссии высшего учебного заведения [Электронный ресурс]. – Режим доступа: https://abit.grsu.by/21-vysshee/normativnye-dokumenty/121-polozhenie-o-priemnoj-komissii-vysshego-uchebnogo-zavedeniya.html/.

[2] Льготы при поступлении в вузы в 2021 году [Электронный ресурс]. – Режим доступа: https://adukar.com/by/news/abiturientu/lgoty-v-vuz.

[3] Руководство по MS SQL Server 2019 [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/sqlserver/.

[4] Администратор базы данных [Электронный ресурс]. – Режим доступа: https://www.profguide.io/Administrator\_bazy\_dannykh.html/.

[5] Обзор основных sql запросов [Электронный ресурс]. – Режим доступа: https://itvdn.com/ru/blog/article/m-sql#1/.

# ПРИЛОЖЕНИЕ А (обязательное) Скрипт генерации базы данных

CREATE DATABASE FOR\_ABITURIENTS;

GO

USE FOR\_ABITURIENTS;

CREATE TABLE GENDERS(

ID INT IDENTITY PRIMARY KEY,

GENDER\_NAME VARCHAR(50) NOT NULL);

CREATE TABLE FAMILY\_STATUS(

ID INT IDENTITY PRIMARY KEY,

STATUS\_NAME VARCHAR(255) NOT NULL);

CREATE TABLE DOCUMENT\_TYPES(

ID INT IDENTITY PRIMARY KEY,

DOC\_NAME VARCHAR(255) NOT NULL);

CREATE TABLE STATUS\_PRIVS(

ID INT IDENTITY PRIMARY KEY,

STATUS\_PRIVS\_NAME VARCHAR(255) NOT NULL);

CREATE TABLE PRIVELEGIES(

ID INT IDENTITY PRIMARY KEY,

PRIV\_NAME VARCHAR(1000) NOT NULL,

STATUS\_PRIV\_ID INT NOT NULL,

STATUS\_ACTIVE INT NOT NULL,

FOREIGN KEY (STATUS\_PRIV\_ID) REFERENCES STATUS\_PRIVS (ID));

CREATE TABLE AB\_STATUS(

ID INT IDENTITY PRIMARY KEY,

ABST\_NAME VARCHAR(255) NOT NULL);

CREATE TABLE FACULTIES(

ID INT IDENTITY PRIMARY KEY,

NAME\_FACULTY VARCHAR(50) NOT NULL,

CORPUS INT);

CREATE TABLE PROFESSORS(

ID INT IDENTITY PRIMARY KEY,

NAME\_PROF VARCHAR(50) NOT NULL,

ID\_FACULTY INT NOT NULL,

LOGIN VARCHAR(50),

FOREIGN KEY (ID\_FACULTY) REFERENCES FACULTIES (ID));

CREATE TABLE TIMETABLE\_TYPES(

ID INT IDENTITY PRIMARY KEY,

NAME\_TYPE VARCHAR(50) NOT NULL);

CREATE TABLE SUBJECTS(

ID INT IDENTITY PRIMARY KEY,

NAME\_SUBJECT VARCHAR(100) NOT NULL);

CREATE TABLE ABITURIENTS(

ID INT IDENTITY PRIMARY KEY,

SURNAME\_R VARCHAR(50) NOT NULL,

NAME\_R VARCHAR(50) NOT NULL,

MIDDLE\_NAME\_R VARCHAR(50),

SURNAME\_L VARCHAR(50) NOT NULL,

NAME\_L VARCHAR(50) NOT NULL,

ID\_GENDER INT NOT NULL,

DATE\_OF\_BIRTH DATE NOT NULL,

ID\_FAMILY\_STATUS INT NOT NULL,

ID\_DOCUMENT\_TYPE INT NOT NULL,

NUMBER\_ID VARCHAR(14) NOT NULL,

SERIAL VARCHAR(2) NOT NULL,

SER\_NUMBER VARCHAR(7) NOT NULL,

ISSUE\_AUTHOR VARCHAR(50) NOT NULL,

DATE\_FORM\_ISSUE DATE NOT NULL,

DATE\_FOR\_ISSUE DATE NOT NULL,

AVER\_EXAM NUMERIC(4,2),

ID\_PRIVILEGES INT NOT NULL,

AVER\_DIPLOM NUMERIC(4,2) NOT NULL,

RESULT\_POINTS NUMERIC(4,2),

ID\_STATUS\_ABITURIENT INT NOT NULL,

GROUP\_EXAM INT,

LOGIN\_ABI VARCHAR(50),

FOREIGN KEY (ID\_GENDER) REFERENCES GENDERS (ID),

FOREIGN KEY (ID\_FAMILY\_STATUS) REFERENCES FAMILY\_STATUS (ID),

FOREIGN KEY (ID\_DOCUMENT\_TYPE) REFERENCES DOCUMENT\_TYPES (ID),

FOREIGN KEY (ID\_PRIVILEGES) REFERENCES PRIVELEGIES (ID),

FOREIGN KEY (ID\_STATUS\_ABITURIENT) REFERENCES AB\_STATUS (ID));

CREATE TABLE EXAM\_LISTS(

ID INT IDENTITY PRIMARY KEY,

ID\_SUBJECT INT NOT NULL,

ID\_ABITURIENT INT NOT NULL,

ID\_PROFESSOR INT NOT NULL,

EXAM\_GRADE NUMERIC(4,2) NOT NULL,

FOREIGN KEY (ID\_SUBJECT) REFERENCES SUBJECTS (ID),

FOREIGN KEY (ID\_ABITURIENT) REFERENCES ABITURIENTS (ID),

FOREIGN KEY (ID\_PROFESSOR) REFERENCES PROFESSORS (ID));

CREATE TABLE FORMS\_STUDY(

ID INT IDENTITY PRIMARY KEY,

NAME\_FORM VARCHAR(100) NOT NULL);

CREATE TABLE SPECIALITIES(

ID INT IDENTITY PRIMARY KEY,

NAME\_SPEC VARCHAR(100) NOT NULL,

ID\_FACULTY INT NOT NULL,

NUM\_STUDENTS INT NOT NULL,

SPECIALIZATION VARCHAR(100) NOT NULL,

ID\_FORM INT NOT NULL,

FOREIGN KEY (ID\_FACULTY) REFERENCES FACULTIES (ID),

FOREIGN KEY (ID\_FORM) REFERENCES FORMS\_STUDY (ID));

CREATE TABLE TIMETABLE(

ID INT IDENTITY PRIMARY KEY,

ID\_TYPE INT NOT NULL,

ID\_SUBJECT INT NOT NULL,

DATE\_OF\_MEET DATETIME NOT NULL,

AUDITORY VARCHAR(10) NOT NULL,

ID\_SPECIALTIES INT NOT NULL,

FOREIGN KEY (ID\_TYPE) REFERENCES TIMETABLE\_TYPES (ID),

FOREIGN KEY (ID\_SUBJECT) REFERENCES SUBJECTS (ID),

FOREIGN KEY (ID\_SPECIALTIES) REFERENCES SPECIALITIES (ID));

CREATE TABLE PRIORITY\_OF\_SPECIALITIES(

ID INT IDENTITY PRIMARY KEY,

ID\_ABITURIENT INT NOT NULL,

ID\_SPECIALTIES INT NOT NULL,

PRIORITY INT NOT NULL,

FOREIGN KEY (ID\_ABITURIENT) REFERENCES ABITURIENTS (ID),

FOREIGN KEY (ID\_SPECIALTIES) REFERENCES SPECIALITIES (ID));

CREATE TABLE STUDENTS(

ID INT IDENTITY PRIMARY KEY,

SURNAME\_R VARCHAR(50) NOT NULL,

NAME\_R VARCHAR(50) NOT NULL,

MIDDLE\_NAME\_R VARCHAR(50),

SURNAME\_L VARCHAR(50) NOT NULL,

NAME\_L VARCHAR(50) NOT NULL,

ID\_GENDER INT NOT NULL,

DATE\_OF\_BIRTH DATE NOT NULL,

ID\_FAMILY\_STATUS INT NOT NULL,

ID\_DOCUMENT\_TYPE INT NOT NULL,

NUMBER\_ID VARCHAR(14) NOT NULL,

SERIAL VARCHAR(2) NOT NULL,

SER\_NUMBER VARCHAR(7) NOT NULL,

ISSUE\_AUTHOR VARCHAR(50) NOT NULL,

DATE\_FORM\_ISSUE DATE NOT NULL,

DATE\_FOR\_ISSUE DATE NOT NULL,

NUM\_GROUP INT NOT NULL,

ID\_SPECIALTIES INT NOT NULL,

ID\_PRIVILEGES INT NOT NULL,

FOREIGN KEY (ID\_GENDER) REFERENCES GENDERS (ID),

FOREIGN KEY (ID\_FAMILY\_STATUS) REFERENCES FAMILY\_STATUS (ID),

FOREIGN KEY (ID\_DOCUMENT\_TYPE) REFERENCES DOCUMENT\_TYPES (ID),

FOREIGN KEY (ID\_SPECIALTIES) REFERENCES SPECIALITIES (ID),

FOREIGN KEY (ID\_PRIVILEGES) REFERENCES PRIVELEGIES (ID));

INSERT GENDERS VALUES ('Мужской');

INSERT GENDERS VALUES ('Женский');

INSERT FAMILY\_STATUS VALUES ('Холост. Детей нет');

INSERT FAMILY\_STATUS VALUES ('Холост. Есть дети');

INSERT FAMILY\_STATUS VALUES ('Женат. Детей нет');

INSERT FAMILY\_STATUS VALUES ('Женат. Есть дети');

INSERT FAMILY\_STATUS VALUES ('Не замужем. Детей нет');

INSERT FAMILY\_STATUS VALUES ('Не замужем. Есть дети');

INSERT FAMILY\_STATUS VALUES ('Замужем. Детей нет');

INSERT FAMILY\_STATUS VALUES ('Замужем. Есть дети');

INSERT DOCUMENT\_TYPES VALUES ('Паспорт');

INSERT DOCUMENT\_TYPES VALUES ('Вид на жительство');

INSERT STATUS\_PRIVS VALUES ('На общих основаниях');

INSERT STATUS\_PRIVS VALUES ('Без вступительных испытаний испытаний');

INSERT STATUS\_PRIVS VALUES ('Вне конкурса');

INSERT STATUS\_PRIVS VALUES ('Преимущестенное право');

INSERT PRIVELEGIES VALUES ('Льгот нет', 1, 1);

INSERT PRIVELEGIES VALUES ('Ребенок-сирота и ребенок, оставшийся без попечения родителей, а также лицо из числа детей-сирот и детей, оставшихся без попечения родителей', 2, 1);

INSERT AB\_STATUS VALUES ('Документы приняты');

INSERT AB\_STATUS VALUES ('Документы не приняты');

INSERT AB\_STATUS VALUES ('Документы отданы');

INSERT AB\_STATUS VALUES ('Зачислен на курс');

INSERT AB\_STATUS VALUES ('Не прошел по конкурсу');

INSERT FORMS\_STUDY VALUES ('Дневная');

INSERT FORMS\_STUDY VALUES ('Заочная');

INSERT FORMS\_STUDY VALUES ('Вечерняя');

INSERT FACULTIES VALUES ('Факультет радиотехники и электроники');

INSERT FACULTIES VALUES ('Факультет компьютерного проектирования');

INSERT FACULTIES VALUES ('Факультет компьютерных технологий');

INSERT SPECIALITIES VALUES ('Профессиональное обучение (информатика)', 1, 30, 'Педагог-программист', 2);

INSERT SPECIALITIES VALUES ('Информационные системы и технологии (в бизнес-менеджменте)', 2, 30, 'Программист. Бизнес-аналитик', 1);

INSERT SPECIALITIES VALUES ('Программное обеспечение информационных технологий', 3, 60, 'Инженер-программист', 2);

INSERT SPECIALITIES VALUES ('Проектирование и производство программно-управляемых электронных средств', 2, 30, 'Инженер-электроник-программист', 2);

INSERT SPECIALITIES VALUES ('Электронные системы безопасности', 2, 30, 'Инженер-проектировщик', 2);

INSERT SPECIALITIES VALUES ('Программируемые мобильные системы', 2, 30, 'Инженер по электронным системам', 1);

INSERT SPECIALITIES VALUES ('Радиотехника (программируемые радиоэлектронные средства)', 1, 30, 'Инженер по радиоэлектронике', 2);

INSERT SPECIALITIES VALUES ('Промышленная электроника', 3, 30, 'Инженер по радиоэлектронике', 3);

INSERT SPECIALITIES VALUES ('Информационные технологии и управление в технических системах', 3, 30, 'Инженер по информационным технологиям и управлению', 3);

INSERT TIMETABLE\_TYPES VALUES ('Консультация');

INSERT TIMETABLE\_TYPES VALUES ('Экзамен');

INSERT SUBJECTS VALUES ('Основы алгоритмизации и программирования');

INSERT SUBJECTS VALUES ('Основы информационных технологий');

INSERT SUBJECTS VALUES ('Охрана труда. Охрана окружающей среды и энергосбережение');

INSERT PROFESSORS VALUES ('Луцик Юрий Александрович', 2);

INSERT PROFESSORS VALUES ('Рак Татьяна Александровна', 2);

INSERT PROFESSORS VALUES ('Оношко Дмитрий Евгеньевич', 2);

INSERT PROFESSORS VALUES ('Лукьянова Ирина Викторовна', 2);

INSERT PROFESSORS VALUES ('Ковальчук Анна Михайловна', 2);

INSERT PROFESSORS VALUES ('Желтко Юлия Юрьевна', 2);

INSERT PROFESSORS VALUES ('Телеш Инна Анатольевна', 1);

INSERT PROFESSORS VALUES ('Рышкель Оксана Станиславовна', 1);

INSERT PROFESSORS VALUES ('Ломонос Ольга Леонидовна', 1);

INSERT PROFESSORS VALUES ('Хлудеев Иван Иванович', 1);

INSERT PROFESSORS VALUES ('Бобровничая Марина Анатольевна', 1);

INSERT PROFESSORS VALUES ('Егоров Владислав Владимирович', 1);

INSERT PROFESSORS VALUES ('Колосов Станислав Васильевич', 3);

INSERT PROFESSORS VALUES ('Матвеенко Владимир Владимирович', 3);

INSERT PROFESSORS VALUES ('Мацкевич Ирина Юрьевна', 3);

INSERT PROFESSORS VALUES ('Павлова Татьяна Григорьевна', 3);

INSERT PROFESSORS VALUES ('Титова Анастасия Владимировна', 3);

INSERT PROFESSORS VALUES ('Кунцевич Ольга Юрьевна', 3);

CREATE ROLE PROFESSORS AUTHORIZATION PROFESSOR;

GRANT SELECT, INSERT, UPDATE, DELETE ON AB\_STATUS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON ABITURIENTS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON DOCUMENT\_TYPES TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON EXAM\_LISTS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON FACULTIES TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON FAMILY\_STATUS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON FORMS\_STUDY TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON GENDERS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON PRIORITY\_OF\_SPECIALITIES TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON PRIVELEGIES TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON PROFESSORS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON SPECIALITIES TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON STATUS\_PRIVS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON STUDENTS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON SUBJECTS TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON TIMETABLE TO ADMIN\_ABI;

GRANT SELECT, INSERT, UPDATE, DELETE ON TIMETABLE\_TYPES TO ADMIN\_ABI;

CREATE LOGIN ABITURIENT WITH PASSWORD = 'Abitur2021';

CREATE USER ABITURIENT FOR LOGIN ABITURIENT;

CREATE ROLE ABITURIENTS AUTHORIZATION ABITURIENT;

CREATE LOGIN COMMISSION WITH PASSWORD = 'AdminAbi2021';

CREATE USER COMMISSION FOR LOGIN COMMISSION;

CREATE ROLE COMMISSIONS AUTHORIZATION COMMISSION;

GRANT UPDATE ON ABITURIENTS TO PROFESSORS;

GRANT UPDATE ON EXAM\_LISTS TO PROFESSORS;

GRANT SELECT ON AB\_STATUS TO ABITURIENTS;

GRANT SELECT ON ABITURIENTS TO ABITURIENTS;

GRANT SELECT ON DOCUMENT\_TYPES TO ABITURIENTS;

GRANT SELECT ON FACULTIES TO ABITURIENTS;

GRANT SELECT ON FAMILY\_STATUS TO ABITURIENTS;

GRANT SELECT ON FORMS\_STUDY TO ABITURIENTS;

GRANT SELECT ON GENDERS TO ABITURIENTS;

GRANT SELECT, INSERT, UPDATE, DELETE ON PRIORITY\_OF\_SPECIALITIES TO ABITURIENTS;

GRANT SELECT ON PRIVELEGIES TO ABITURIENTS;

GRANT SELECT ON SPECIALITIES TO ABITURIENTS;

GRANT SELECT ON SUBJECTS TO ABITURIENTS;

GRANT SELECT ON TIMETABLE TO ABITURIENTS;

GRANT SELECT ON TIMETABLE\_TYPES TO ABITURIENTS;

CREATE OR ALTER PROCEDURE SET\_EXAM\_GRADE (@id\_abi INT)

AS

SET NOCOUNT ON;

DECLARE @avg\_abi NUMERIC(4,2);

DECLARE exams CURSOR FOR

SELECT AVG(EXAM\_GRADE) FROM EXAM\_LISTS WHERE ID\_ABITURIENT = @id\_abi GROUP BY ID\_ABITURIENT;

OPEN exams;

FETCH NEXT FROM exams INTO @avg\_abi;

WHILE @@FETCH\_STATUS = 0

BEGIN

UPDATE ABITURIENTS

SET AVER\_EXAM = @avg\_abi

WHERE ID = @id\_abi;

FETCH NEXT FROM exams INTO @avg\_abi;

END;

CLOSE exams;

DEALLOCATE exams;

CREATE OR ALTER PROCEDURE SET\_RESULT\_POINT (@id\_abi INT)

AS

SET NOCOUNT ON;

DECLARE @id\_abi INT; --ID\_ABITURIENT

DECLARE @avg\_abi NUMERIC(4,2); --AVG(RESULT)

DECLARE abiturient CURSOR FOR

SELECT ID, AVER\_EXAM + AVER\_DIPLOM FROM ABITURIENTS;

OPEN abiturient;

FETCH NEXT FROM abiturient INTO @id\_abi, @avg\_abi;

WHILE @@FETCH\_STATUS = 0

BEGIN

UPDATE ABITURIENTS

SET RESULT\_POINTS = @avg\_abi

WHERE ID = @id\_abi;

FETCH NEXT FROM abiturient INTO @id\_abi, @avg\_abi;

END;

CLOSE abiturient;

DEALLOCATE abiturient;

CREATE OR ALTER PROCEDURE SET\_TIMETABLE

AS

DECLARE @id\_cab VARCHAR(7);

DECLARE @id\_start\_exam DATETIME = '2021-07-10 09:00:00';

DECLARE @id\_middle\_exam DATETIME = '2021-07-14 09:00:00';

DECLARE @id\_finish\_exam DATETIME = '2021-07-18 09:00:00';

DECLARE @id\_meet VARCHAR(50);

DECLARE @DATE\_MEET DATETIME;

DECLARE @ID INT;

DECLARE @ID\_CORP INT;

DECLARE @id\_sub INT = 1;

DECLARE specilities CURSOR FOR

SELECT SPECIALITIES.ID FROM SPECIALITIES;

OPEN specilities;

FETCH NEXT FROM specilities INTO @ID;

WHILE @ID <= 9

BEGIN

SELECT @ID\_CORP = (SELECT FACULTIES.CORPUS FROM SPECIALITIES, FACULTIES

WHERE FACULTIES.ID = SPECIALITIES.ID\_FACULTY AND SPECIALITIES.ID = @ID);

INSERT INTO TIMETABLE (ID\_TYPE, ID\_SUBJECT, DATE\_OF\_MEET, AUDITORY, ID\_SPECIALTIES)

VALUES (1, 1, @id\_start\_exam, CONVERT(VARCHAR(20),@ID\_CORP,0) + '-108', @ID);

SELECT @ID = @ID + 1;

END;

SELECT @ID = 1;

WHILE @ID <= 9

BEGIN

SELECT @ID\_CORP = (SELECT FACULTIES.CORPUS FROM SPECIALITIES, FACULTIES

WHERE FACULTIES.ID = SPECIALITIES.ID\_FACULTY AND SPECIALITIES.ID = @ID);

INSERT INTO TIMETABLE (ID\_TYPE, ID\_SUBJECT, DATE\_OF\_MEET, AUDITORY, ID\_SPECIALTIES)

VALUES (1, 2, @id\_middle\_exam, CONVERT(VARCHAR(20),@ID\_CORP,0) + '-108', @ID);

SELECT @ID = @ID + 1;

END;

SELECT @ID = 1;

WHILE @ID <= 9

BEGIN

SELECT @ID\_CORP = (SELECT FACULTIES.CORPUS FROM SPECIALITIES, FACULTIES

WHERE FACULTIES.ID = SPECIALITIES.ID\_FACULTY AND SPECIALITIES.ID = @ID);

INSERT INTO TIMETABLE (ID\_TYPE, ID\_SUBJECT, DATE\_OF\_MEET, AUDITORY, ID\_SPECIALTIES)

VALUES (1, 3, @id\_finish\_exam, CONVERT(VARCHAR(20),@ID\_CORP,0) + '-108', @ID);

SELECT @ID = @ID + 1;

END;

SELECT @ID = 1;

WHILE @ID <= 9

BEGIN

WHILE @id\_sub <= 3 AND @ID <= 9

BEGIN

SELECT @ID\_CORP = (SELECT FACULTIES.CORPUS FROM SPECIALITIES, FACULTIES

WHERE FACULTIES.ID = SPECIALITIES.ID\_FACULTY AND SPECIALITIES.ID = @ID);

SELECT @DATE\_MEET = (SELECT DATEADD(DAY, 1, DATE\_OF\_MEET) FROM TIMETABLE

WHERE ID\_SUBJECT = @id\_sub AND ID\_TYPE = 1 AND ID\_SPECIALTIES = @ID);

INSERT INTO TIMETABLE (ID\_TYPE, ID\_SUBJECT, DATE\_OF\_MEET, AUDITORY, ID\_SPECIALTIES)

VALUES (2, @id\_sub, @DATE\_MEET, CONVERT(VARCHAR(20),@ID\_CORP,0) + '-308', @ID);

SELECT @id\_sub = @id\_sub + 1;

END;

SELECT @ID = @ID + 1;

SELECT @id\_sub = 1;

END;

CLOSE specilities;

DEALLOCATE specilities;

CREATE OR ALTER TRIGGER SET\_GROUP\_EXAM

ON PRIORITY\_OF\_SPECIALITIES

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @ID VARCHAR(20);

SELECT @ID = (SELECT ID\_ABITURIENT FROM inserted);

PRINT (@ID);

UPDATE ABITURIENTS SET GROUP\_EXAM =

(SELECT CONVERT(VARCHAR(20),CORPUS,0) + CONVERT(VARCHAR(20),CORPUS,0) + SUBSTRING(CONVERT(VARCHAR(20),SYSDATETIME(),3),7,2)

FROM FACULTIES, SPECIALITIES, PRIORITY\_OF\_SPECIALITIES

WHERE PRIORITY\_OF\_SPECIALITIES.ID\_ABITURIENT = ABITURIENTS.ID AND

PRIORITY\_OF\_SPECIALITIES.PRIORITY = 1 AND

PRIORITY\_OF\_SPECIALITIES.ID\_SPECIALTIES = SPECIALITIES.ID AND

SPECIALITIES.ID\_FACULTY = FACULTIES.ID AND

ABITURIENTS.ID = @ID)

WHERE ABITURIENTS.ID = @ID;

END;

CREATE OR ALTER TRIGGER SET\_AVER\_EXAM

ON EXAM\_LISTS

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @ID\_SUB INT;

SELECT @ID\_SUB = (SELECT ID\_SUBJECT FROM inserted);

IF @ID\_SUB = 3

BEGIN

DECLARE @ID INT;

SELECT @ID = (SELECT ID\_ABITURIENT FROM inserted);

EXEC SET\_EXAM\_GRADE @ID;

EXEC SET\_RESULT\_POINT @ID;

END

END;

# ПРИЛОЖЕНИЕ Б (обязательное) Листинг программного кода

class InProf(QtWidgets.QDialog):

def \_\_init\_\_(self, name, password, id\_sub, parent=None):

super(InProf, self).\_\_init\_\_(parent)

self.name = name

self.password = password

self.id\_sub = id\_sub

self.initUI()

def initUI(self):

self.verticalLayout = QtWidgets.QVBoxLayout(self)

self.verticalLayout.setObjectName("verticalLayout")

self.setFixedSize(1450, 900)

self.setWindowTitle("Личный кабинет преподавателя")

self.scroll = QScrollArea(self)

self.widget = QWidget(self)

self.vbox = QVBoxLayout(self)

server = "LAPTOP-N2VDG922\SQLEXPRESS"

database = "FOR\_ABITURIENTS"

self.cnxn = pyodbc.connect("Driver={SQL Server Native Client 11.0};"

"Server=" + server + ";"

"Database=" + database + ";"

"username=" + self.name + ";"

"password=" + self.password + ";" "Trusted\_Connection=yes;")

self.query = "-- {}\n\n-- Made in Python".format(datetime.now().strftime("%d/%m/%Y"))

cursor = self.cnxn.cursor()

cursor.fast\_executemany = True

query = "SELECT CONVERT(VARCHAR(20),ID,0) + ' ФИО: ' + SURNAME\_R + ' ' + NAME\_R + ' ' + MIDDLE\_NAME\_R + ' Идентификационный номер: ' + NUMBER\_ID FROM ABITURIENTS WHERE ID\_STATUS\_ABITURIENT = 1 ORDER BY SURNAME\_R ASC"

cursor.execute(query)

row = cursor.fetchone()

while row:

self.getText(row[0])

row = cursor.fetchone()

query = "SELECT 'ID: ' +CONVERT(VARCHAR(20),ABITURIENTS.ID,0) + ' ФИО: ' + SURNAME\_R + ' ' + NAME\_R + ' ' + MIDDLE\_NAME\_R + ' Идентификационный номер: ' + NUMBER\_ID, CONVERT(VARCHAR(20),EXAM\_GRADE,0) FROM ABITURIENTS, EXAM\_LISTS WHERE ABITURIENTS.ID = EXAM\_LISTS.ID\_ABITURIENT AND ID\_STATUS\_ABITURIENT = 1 ORDER BY SURNAME\_R ASC"

cursor.execute(query)

row = cursor.fetchone()

while row:

label = QLabel(row[0])

self.vbox.addWidget(label)

textEdit = QTextEdit(row[1])

self.vbox.addWidget(textEdit)

row = cursor.fetchone()

pushButton\_1 = QtWidgets.QPushButton(self)

self.vbox.addWidget(pushButton\_1)

pushButton\_1.setText("Отмена")

pushButton\_1.clicked.connect(self.btnClosed)

self.widget.setLayout(self.vbox)

self.scroll.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

self.scroll.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)

self.scroll.setWidgetResizable(True)

self.scroll.setWidget(self.widget)

self.verticalLayout.addWidget(self.scroll)

self.setGeometry(300, 80, 1000, 900)

self.show()

def btnClosed(self):

self.close()

def getText(self, row):

server = "LAPTOP-N2VDG922\SQLEXPRESS"

database = "FOR\_ABITURIENTS"

self.cnxn = pyodbc.connect("Driver={SQL Server Native Client 11.0};"

"Server=" + server + ";"

"Database=" + database + ";"

"username=" + self.name + ";"

"password=" + self.password + ";" "Trusted\_Connection=yes;")

cursor = self.cnxn.cursor()

cursor.fast\_executemany = True

try:

text, okPressed = QInputDialog.getText(self, "Выставить отметку за вступительный экзамен", row+" Отметка:",

QLineEdit.Normal, "")

if okPressed and text != '':

query = "INSERT EXAM\_LISTS VALUES (CONVERT(INT,"+ self.id\_sub + ",0),(SELECT ID FROM ABITURIENTS WHERE ID = SUBSTRING ('"+ row +"', CHARINDEX('RANDOM','"+ row +"'), CHARINDEX('ФИО','"+ row +"')-1)),(SELECT ID FROM PROFESSORS WHERE LOGIN = '"+self.name+"'), "+text+")"

cursor.execute(query)

self.cnxn.commit()

except Exception as e:

print(e)

class InAbi(QtWidgets.QDialog):

def \_\_init\_\_(self, name, password, parent=None):

super(InAbi, self).\_\_init\_\_(parent)

self.name = name

self.password = password

self.initUI()

def initUI(self):

self.verticalLayout = QtWidgets.QVBoxLayout(self)

self.verticalLayout.setObjectName("verticalLayout")

self.setFixedSize(1450, 900)

self.setWindowTitle("Личный кабинет абитуриента")

self.scroll = QScrollArea(self)

self.widget = QWidget(self)

self.vbox = QVBoxLayout(self)

server = "LAPTOP-N2VDG922\SQLEXPRESS"

database = "FOR\_ABITURIENTS"

self.cnxn = pyodbc.connect("Driver={SQL Server Native Client 11.0};"

"Server=" + server + ";"

"Database=" + database + ";"

"username=" + self.name + ";"

"password=" + self.password + ";" "Trusted\_Connection=yes;")

self.query = "-- {}\n\n-- Made in Python".format(datetime.now().strftime("%d/%m/%Y"))

cursor = self.cnxn.cursor()

cursor.fast\_executemany = True

query = "SELECT SURNAME\_R,NAME\_R,MIDDLE\_NAME\_R,SURNAME\_L,NAME\_L,CONVERT(VARCHAR(20),DATE\_OF\_BIRTH,0), CONVERT(VARCHAR(20),AVER\_DIPLOM,0) FROM ABITURIENTS WHERE LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

object = QLabel("Фамилия абитуриента на русском языке")

self.vbox.addWidget(object)

self.textEdit = QtWidgets.QTextEdit(self)

self.textEdit.setText(row[0])

self.textEdit.setReadOnly(True)

self.vbox.addWidget(self.textEdit)

object\_1 = QLabel("Имя абитуриента на русском языке")

self.vbox.addWidget(object\_1)

self.textEdit\_1 = QtWidgets.QTextEdit(self)

self.textEdit\_1.setText(row[1])

self.textEdit\_1.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_1)

object\_2 = QLabel("Отчество абитуриента на русском языке")

self.vbox.addWidget(object\_2)

self.textEdit\_2 = QtWidgets.QTextEdit(self)

self.textEdit\_2.setText(row[2])

self.textEdit\_2.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_2)

object\_3 = QLabel("Фамилия абитуриента на английском языке")

self.vbox.addWidget(object\_3)

self.textEdit\_3 = QtWidgets.QTextEdit(self)

self.textEdit\_3.setText(row[3])

self.textEdit\_3.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_3)

object\_4 = QLabel("Имя абитуриента на английском языке")

self.vbox.addWidget(object\_4)

self.textEdit\_4 = QtWidgets.QTextEdit(self)

self.textEdit\_4.setText(row[4])

self.textEdit\_4.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_4)

object\_5 = QLabel("Дата рождения абитуриента")

self.vbox.addWidget(object\_5)

self.textEdit\_5 = QtWidgets.QTextEdit(self)

self.textEdit\_5.setText(row[5])

self.textEdit\_5.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_5)

object\_9 = QLabel("Средний балл атестата")

self.textEdit\_9 = QtWidgets.QTextEdit(self)

self.textEdit\_9.setText(row[6])

self.textEdit\_9.setReadOnly(True)

row = cursor.fetchone()

query = "SELECT STATUS\_NAME FROM FAMILY\_STATUS, ABITURIENTS WHERE FAMILY\_STATUS.ID = ABITURIENTS.ID\_FAMILY\_STATUS AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

object\_6 = QLabel("Семейное положение абитуриента")

self.vbox.addWidget(object\_6)

self.textEdit\_6 = QtWidgets.QTextEdit(self)

self.textEdit\_6.setText(row[0])

self.textEdit\_6.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_6)

row = cursor.fetchone()

query = "SELECT GENDER\_NAME FROM GENDERS, ABITURIENTS WHERE GENDERS.ID = ABITURIENTS.ID\_GENDER AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

object\_7 = QLabel("Пол абитуриента")

self.vbox.addWidget(object\_7)

self.textEdit\_7 = QtWidgets.QTextEdit(self)

self.textEdit\_7.setText(row[0])

self.textEdit\_7.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_7)

row = cursor.fetchone()

query = "SELECT DOC\_NAME + ' серия: '+ SERIAL + ' номер: ' + SER\_NUMBER + ' идентификационный номер: ' + NUMBER\_ID + ' выданный: ' + ISSUE\_AUTHOR + ' срок действия: с ' + CONVERT(VARCHAR(20),DATE\_FORM\_ISSUE,0) + ' по ' + CONVERT(VARCHAR(20),DATE\_FOR\_ISSUE,0) FROM DOCUMENT\_TYPES, ABITURIENTS WHERE DOCUMENT\_TYPES.ID = ABITURIENTS.ID\_DOCUMENT\_TYPE AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

object\_8 = QLabel("Документ, удостоверяющий личность абитуриента")

self.vbox.addWidget(object\_8)

self.textEdit\_8 = QtWidgets.QTextEdit(self)

self.textEdit\_8.setText(row[0])

self.textEdit\_8.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_8)

row = cursor.fetchone()

query = "SELECT PRIV\_NAME FROM PRIVELEGIES, ABITURIENTS WHERE PRIVELEGIES.ID = ABITURIENTS.ID\_PRIVILEGES AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

object\_10 = QLabel("Льготы абитуриента")

self.vbox.addWidget(object\_10)

self.textEdit\_10 = QtWidgets.QTextEdit(self)

self.textEdit\_10.setText(row[0])

self.textEdit\_10.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_10)

self.vbox.addWidget(object\_9)

self.vbox.addWidget(self.textEdit\_9)

row = cursor.fetchone()

query = "SELECT NAME\_SPEC FROM SPECIALITIES, PRIORITY\_OF\_SPECIALITIES, ABITURIENTS WHERE PRIORITY\_OF\_SPECIALITIES.ID\_ABITURIENT = ABITURIENTS.ID AND PRIORITY\_OF\_SPECIALITIES.PRIORITY = 1 AND PRIORITY\_OF\_SPECIALITIES.ID\_SPECIALTIES = SPECIALITIES.ID AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

query = "SELECT CONVERT(VARCHAR(20),GROUP\_EXAM,0) FROM ABITURIENTS WHERE LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

object\_14 = QLabel("РАСПИСАНИЕ ВСТУПИТЕЛЬНЫХ ЭКЗАМЕНОВ ГРУППА - " + row[0])

self.vbox.addWidget(object\_14)

row = cursor.fetchone()

object\_15 = QLabel("Основы алгоритмизации и программирования:")

self.vbox.addWidget(object\_15)

query = "SELECT DISTINCT 'Консультация - ' + CONVERT(VARCHAR(20),DATE\_OF\_MEET,0) + ' в аудитории - ' + CONVERT(VARCHAR(20),AUDITORY,0) FROM TIMETABLE, PRIORITY\_OF\_SPECIALITIES, SPECIALITIES, ABITURIENTS WHERE PRIORITY\_OF\_SPECIALITIES.ID\_ABITURIENT = ABITURIENTS.ID AND PRIORITY\_OF\_SPECIALITIES.PRIORITY = 1 AND PRIORITY\_OF\_SPECIALITIES.ID\_SPECIALTIES = SPECIALITIES.ID AND PRIORITY\_OF\_SPECIALITIES.ID\_SPECIALTIES = TIMETABLE.ID\_SPECIALTIES AND TIMETABLE.ID\_TYPE = 1 AND TIMETABLE.ID\_SUBJECT = 1 AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

self.textEdit\_15 = QtWidgets.QTextEdit(self)

self.textEdit\_15.setText(row[0])

self.textEdit\_15.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_15)

row = cursor.fetchone()

query = "SELECT DISTINCT 'Экзамен - ' + CONVERT(VARCHAR(20),DATE\_OF\_MEET,0) + ' в аудитории - ' + CONVERT(VARCHAR(20),AUDITORY,0) FROM TIMETABLE, PRIORITY\_OF\_SPECIALITIES, SPECIALITIES, ABITURIENTS WHERE PRIORITY\_OF\_SPECIALITIES.ID\_ABITURIENT = ABITURIENTS.ID AND PRIORITY\_OF\_SPECIALITIES.PRIORITY = 1 AND PRIORITY\_OF\_SPECIALITIES.ID\_SPECIALTIES = SPECIALITIES.ID AND PRIORITY\_OF\_SPECIALITIES.ID\_SPECIALTIES = TIMETABLE.ID\_SPECIALTIES AND TIMETABLE.ID\_TYPE = 2 AND TIMETABLE.ID\_SUBJECT = 1 AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

self.textEdit\_16 = QtWidgets.QTextEdit(self)

self.textEdit\_16.setText(row[0])

self.textEdit\_16.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_16)

row = cursor.fetchone()

object\_18 = QLabel("РЕЗУЛЬТАТЫ ВСТУПИТЕЛЬНЫХ ЭКЗАМЕНОВ:")

self.vbox.addWidget(object\_18)

object\_19 = QLabel("Основы алгоритмизации и программирования:")

self.vbox.addWidget(object\_19)

query = "SELECT CONVERT(VARCHAR(20),ISNULL(EXAM\_GRADE, 0),0) FROM EXAM\_LISTS, ABITURIENTS WHERE EXAM\_LISTS.ID\_SUBJECT = 1 AND EXAM\_LISTS.ID\_ABITURIENT = ABITURIENTS.ID AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

self.textEdit\_21 = QtWidgets.QTextEdit(self)

self.textEdit\_21.setText(row[0])

self.textEdit\_21.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_21)

row = cursor.fetchone()

object\_22 = QLabel("ИТОГОВЫЙ РЕЗУЛЬТАТ:")

self.vbox.addWidget(object\_22)

query = "SELECT CONVERT(VARCHAR(20),ISNULL(RESULT\_POINTS, 0),0) FROM ABITURIENTS WHERE LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

self.textEdit\_24 = QtWidgets.QTextEdit(self)

self.textEdit\_24.setText(row[0])

self.textEdit\_24.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_24)

row = cursor.fetchone()

object\_23 = QLabel("СТАТУС АБИТУРИЕНТА:")

self.vbox.addWidget(object\_23)

query = "SELECT ABST\_NAME FROM AB\_STATUS, ABITURIENTS WHERE AB\_STATUS.ID = ABITURIENTS.ID\_STATUS\_ABITURIENT AND LOGIN\_ABI = '" + self.name + "'"

cursor.execute(query)

row = cursor.fetchone()

while row:

self.textEdit\_25 = QtWidgets.QTextEdit(self)

self.textEdit\_25.setText(row[0])

self.textEdit\_25.setReadOnly(True)

self.vbox.addWidget(self.textEdit\_25)

row = cursor.fetchone()

self.widget.setLayout(self.vbox)

self.scroll.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

self.scroll.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)

self.scroll.setWidgetResizable(True)

self.scroll.setWidget(self.widget)

self.verticalLayout.addWidget(self.scroll)

self.setGeometry(300, 80, 1000, 900)

self.show()

class OpenZayav(QtWidgets.QDialog):

def \_\_init\_\_(self, parent=None):

super(OpenZayav, self).\_\_init\_\_(parent)

self.initUI()

def initUI(self):

self.verticalLayout = QtWidgets.QVBoxLayout(self)

self.verticalLayout.setObjectName("verticalLayout")

self.setFixedSize(1450, 900)

self.scroll = QScrollArea(self)

self.widget = QWidget(self)

self.vbox = QVBoxLayout(self)

object = QLabel("Введите фамилию на русском языке \*")

self.vbox.addWidget(object)

self.textEdit = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit)

object\_1 = QLabel("Введите имя на русском языке \*")

self.vbox.addWidget(object\_1)

self.textEdit\_1 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_1)

object\_2 = QLabel("Введите отчество на русском языке")

self.vbox.addWidget(object\_2)

self.textEdit\_2 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_2)

object\_3 = QLabel("Введите фамилию на английском языке \*")

self.vbox.addWidget(object\_3)

self.textEdit\_3 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_3)

object\_4 = QLabel("Введите имя на английском языке \*")

self.vbox.addWidget(object\_4)

self.textEdit\_4 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_4)

object\_5 = QLabel("Введите дату рождения в формате YYYY-MM-DD \*")

self.vbox.addWidget(object\_5)

self.textEdit\_5 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_5)

server = "LAPTOP-N2VDG922\SQLEXPRESS"

name = "ADMIN\_ABI"

passw = "AdminAbi2021"

database = "FOR\_ABITURIENTS"

self.cnxn = pyodbc.connect("Driver={SQL Server Native Client 11.0};"

"Server=" + server + ";"

"Database=" + database + ";"

"username=" + name + ";"

"password=" + passw + ";" "Trusted\_Connection=yes;")

self.query = "-- {}\n\n-- Made in Python".format(datetime.now().strftime("%d/%m/%Y"))

cursor = self.cnxn.cursor()

cursor.fast\_executemany = True

self.comboBox\_3 = QtWidgets.QComboBox(self)

self.comboBox\_3.clear()

query = "SELECT DOC\_NAME FROM DOCUMENT\_TYPES"

cursor.execute(query)

for i in cursor:

self.comboBox\_3.addItems(i)

# self.cnxn.commit()

object\_15 = QLabel("Выберите тип документа, удостоверяющий личность")

self.vbox.addWidget(object\_15)

self.vbox.addWidget(self.comboBox\_3)

object\_6 = QLabel("Введите номер документа, удостоверяющий личность \*")

self.vbox.addWidget(object\_6)

self.textEdit\_6 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_6)

object\_7 = QLabel("Введите серию документа, удостоверяющий личность \*")

self.vbox.addWidget(object\_7)

self.textEdit\_7 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_7)

object\_8 = QLabel("Введите идентификационный номер документа, удостоверяющий личность \*")

self.vbox.addWidget(object\_8)

self.textEdit\_8 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_8)

object\_9 = QLabel("Введите дату выдачи документа, удостоверяющий личность в формате YYYY-MM-DD \*")

self.vbox.addWidget(object\_9)

self.textEdit\_9 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_9)

object\_10 = QLabel("Введите дату окончания действия документа, удостоверяющий личность в формате YYYY-MM-DD \*")

self.vbox.addWidget(object\_10)

self.textEdit\_10 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_10)

object\_14 = QLabel("Введите орган выдавший документ, удостоверяющий личность \*")

self.vbox.addWidget(object\_14)

self.textEdit\_11 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_11)

self.comboBox = QtWidgets.QComboBox(self)

self.comboBox.clear()

query = "SELECT GENDER\_NAME FROM GENDERS"

cursor.execute(query)

for i in cursor:

self.comboBox.addItems(i)

# self.cnxn.commit()

object\_11 = QLabel("Выберите пол")

self.vbox.addWidget(object\_11)

self.vbox.addWidget(self.comboBox)

self.comboBox\_1 = QtWidgets.QComboBox(self)

self.comboBox\_1.clear()

query = "SELECT PRIV\_NAME FROM PRIVELEGIES WHERE STATUS\_ACTIVE = 1"

cursor.execute(query)

for i in cursor:

self.comboBox\_1.addItems(i)

# self.cnxn.commit()

object\_12 = QLabel("Выберите льготы")

self.vbox.addWidget(object\_12)

self.vbox.addWidget(self.comboBox\_1)

self.comboBox\_2 = QtWidgets.QComboBox(self)

self.comboBox\_2.clear()

query = "SELECT STATUS\_NAME FROM FAMILY\_STATUS"

cursor.execute(query)

for i in cursor:

self.comboBox\_2.addItems(i)

object\_13 = QLabel("Выберите семейное положение")

self.vbox.addWidget(object\_13)

self.vbox.addWidget(self.comboBox\_2)

object\_16 = QLabel("Введите средний балл за диплом в формате NN.NN \*")

self.vbox.addWidget(object\_16)

self.textEdit\_12 = QtWidgets.QTextEdit(self)

self.vbox.addWidget(self.textEdit\_12)

self.comboBox\_4 = QtWidgets.QComboBox(self)

self.comboBox\_4.clear()

query = "SELECT NAME\_SPEC + ' специализация - ' + SPECIALIZATION + ' форма обучения - ' + (SELECT NAME\_FORM FROM FORMS\_STUDY WHERE ID = SPECIALITIES.ID\_FORM) FROM SPECIALITIES"

cursor.execute(query)

for i in cursor:

self.comboBox\_4.addItems(i)

object\_17 = QLabel("Выберите специальность с приоритетом 1")

self.vbox.addWidget(object\_17)

self.vbox.addWidget(self.comboBox\_4)

self.comboBox\_5 = QtWidgets.QComboBox(self)

self.comboBox\_5.clear()

query = "SELECT NAME\_SPEC + ' специализация - ' + SPECIALIZATION + ' форма обучения - ' + (SELECT NAME\_FORM FROM FORMS\_STUDY WHERE ID = SPECIALITIES.ID\_FORM) FROM SPECIALITIES"

cursor.execute(query)

for i in cursor:

self.comboBox\_5.addItems(i)

object\_18 = QLabel("Выберите специальность с приоритетом 2")

self.vbox.addWidget(object\_18)

self.vbox.addWidget(self.comboBox\_5)

self.comboBox\_6 = QtWidgets.QComboBox(self)

self.comboBox\_6.clear()

query = "SELECT NAME\_SPEC + ' специализация - ' + SPECIALIZATION + ' форма обучения - ' + (SELECT NAME\_FORM FROM FORMS\_STUDY WHERE ID = SPECIALITIES.ID\_FORM) FROM SPECIALITIES"

cursor.execute(query)

for i in cursor:

self.comboBox\_6.addItems(i)

object\_19 = QLabel("Выберите специальность с приоритетом 3")

self.setWindowTitle('Подача заявления')

self.show()

def CreateUs(self):

try:

cursor = self.cnxn.cursor()

cursor.fast\_executemany = True

query = "INSERT INTO ABITURIENTS (SURNAME\_R, NAME\_R, MIDDLE\_NAME\_R," \

" SURNAME\_L, NAME\_L, ID\_GENDER, DATE\_OF\_BIRTH, ID\_FAMILY\_STATUS," \

" ID\_DOCUMENT\_TYPE, NUMBER\_ID, SERIAL, SER\_NUMBER, ISSUE\_AUTHOR," \

" DATE\_FORM\_ISSUE, DATE\_FOR\_ISSUE, ID\_PRIVILEGES, AVER\_DIPLOM," \

"ID\_STATUS\_ABITURIENT, LOGIN\_ABI) " \

"VALUES (" + "'" + self.textEdit.toPlainText() + "', " \

+ "'" + self.textEdit\_1.toPlainText() + "', " \

+ "'" + self.textEdit\_2.toPlainText() + "', " \

+ "'" + self.textEdit\_3.toPlainText() + "', " \

+ "'" + self.textEdit\_4.toPlainText() + "', " \

+ "(SELECT ID FROM GENDERS WHERE GENDER\_NAME ='" + self.comboBox.currentText() + "'), " \

+ "'" + self.textEdit\_5.toPlainText() + "', " \

+ "(SELECT ID FROM FAMILY\_STATUS WHERE STATUS\_NAME ='" + self.comboBox\_2.currentText() + "'), " \

+ "(SELECT ID FROM DOCUMENT\_TYPES WHERE DOC\_NAME ='" + self.comboBox\_3.currentText() + "'), " \

+ "'" + self.textEdit\_8.toPlainText() + "', " \

+ "'" + self.textEdit\_7.toPlainText() + "', " \

+ "'" + self.textEdit\_6.toPlainText() + "', " \

+ "'" + self.textEdit\_11.toPlainText() + "', " \

+ "'" + self.textEdit\_9.toPlainText() + "', " \

+ "'" + self.textEdit\_10.toPlainText() + "', " \

+ "(SELECT ID FROM PRIVELEGIES WHERE PRIV\_NAME ='" + self.comboBox\_1.currentText() + "'), " \

+ "'" + self.textEdit\_12.toPlainText() + "', " \

+ "1," \

+ "'" + self.textEdit\_3.toPlainText() + "' " \

+ ")"

cursor.execute(query)

self.cnxn.commit()

query = "INSERT PRIORITY\_OF\_SPECIALITIES values ("+\

"(SELECT ID FROM ABITURIENTS WHERE LOGIN\_ABI = "+ "'" + self.textEdit\_3.toPlainText() + "') "+",(SELECT ID FROM SPECIALITIES WHERE NAME\_SPEC = SUBSTRING ('"+ self.comboBox\_4.currentText() + "', "+\

"CHARINDEX('RANDOM','" + self.comboBox\_4.currentText() + "'), "+\

"CHARINDEX('специализация','" + self.comboBox\_4.currentText() + "')-1)),1)"

cursor.execute(query)

self.cnxn.commit()

query = "INSERT PRIORITY\_OF\_SPECIALITIES values (" + \

"(SELECT ID FROM ABITURIENTS WHERE LOGIN\_ABI = " + "'" + self.textEdit\_3.toPlainText() + "') " + ",(SELECT ID FROM SPECIALITIES WHERE NAME\_SPEC = SUBSTRING ('" + self.comboBox\_4.currentText() + "', " + \

"CHARINDEX('RANDOM','" + self.comboBox\_4.currentText() + "'), " + \

"CHARINDEX('специализация','" + self.comboBox\_4.currentText() + "')-1)),2)"

cursor.execute(query)

self.cnxn.commit()

query = "INSERT PRIORITY\_OF\_SPECIALITIES values (" + \

"(SELECT ID FROM ABITURIENTS WHERE LOGIN\_ABI = " + "'" + self.textEdit\_3.toPlainText() + "') " + ",(SELECT ID FROM SPECIALITIES WHERE NAME\_SPEC = SUBSTRING ('" + self.comboBox\_4.currentText() + "', " + \

"CHARINDEX('RANDOM','" + self.comboBox\_4.currentText() + "'), " + \

"CHARINDEX('специализация','" + self.comboBox\_4.currentText() + "')-1)),3)"

cursor.execute(query)

self.cnxn.commit()

user = self.textEdit\_3.toPlainText()

pas = user + "2021Abi"

query = "CREATE LOGIN " + user + " WITH PASSWORD = '" + pas +"'"

cursor.execute(query)

self.cnxn.commit()

query = "CREATE USER " + user + " FOR LOGIN " + user

cursor.execute(query)

self.cnxn.commit()

query = "ALTER ROLE ABITURIENTS ADD MEMBER " + user

cursor.execute(query)

self.cnxn.commit()

ctypes.windll.user32.MessageBoxW(0, "Ваши документы приняты и будут рассмотрены. \n"

"Ваш логин - " + user + ".\nВаш пароль - " + pas +

".\nВ случае возникновения вопросов, обратитесь в приемную коммиссию",

"Подача заявления в университет", 1)

self.close()

except Exception as e:

ctypes.windll.user32.MessageBoxW(0, "Произошла ошибка! Проверьте все ли поля с \* заполнены или обратитесь в службу поддержки", "Подача заявления в университет", 1)

def btnClosed(self):

self.close()

# ПРИЛОЖЕНИЕ В (обязательное) Отчет о проверке на заимствования



# ПРИЛОЖЕНИЕ Г (обязательное) Ведомость курсового проекта