ГБПОУ Республики Марий Эл «МРМТ»

Допущен к защите

Зам. директора по УМР

\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_ 2024 г.

**ДИПЛОМНЫЙ ПРОЕКТ**

разработка программного обеспечения для учета абитуриентов техникума

Рецензент Разработчик

\_\_\_\_\_\_\_\_\_\_\_ Г. Н. Гайфутдинова Студентка группы П-41

\_\_\_\_\_\_\_\_\_\_ Е.П. Валиев

Нормоконтроль Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_ Л.С. Лебедева \_\_\_\_\_\_\_\_А.Н. Савинов

Оценка Экзаменационной комиссии по защите\_\_\_\_\_\_\_\_

Председатель ГЭК\_\_\_\_\_\_\_\_\_\_\_\_

Йошкар-Ола 2024

ГБПОУ Республики Марий Эл «Марийский радиомеханический техникум»

УТВЕРЖДАЮ

Зам. директора по УМР

\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_2024 г.

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ ДИПЛОМНОГО ПРОЕКТА

|  |  |  |  |
| --- | --- | --- | --- |
| Студенту группы | *П–41* | специальность | *09.02.07* |
| *Валиев Евгений Павлович* | | | |
| (фамилия имя отчество) | | | |

|  |  |
| --- | --- |
| Тема дипломного проекта | *Разработка программного обеспечения для учета* |
| *абитуриентов техникума* | |

|  |  |
| --- | --- |
| Исходные данные | *Операционная система Microsoft Windows, среда разработки* |
| *Visual Studio, язык программирования C#, система управления базами данных* | |
| *MySql* | |

Содержание дипломной работы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Введение |  | | | |
| Теоретический раздел | | | *Понятие программного обеспечения, понятие базы данных* | |
|  | | | | |
| Аналитический раздел | | | *Создание диаграммы прецендентов, выбор средств* | |
| *разработки информационной системы* | | | | |
| Исследовательский раздел | | | | *Технической задание, разработка инфологической* |
| *модели БД, разработка даталогической модели БД, разработка базы данных,* | | | | |
| *разработка интерфейса пользователя, тестирование программного обеспечения* | | | | |
| Заключение | |  | | |

|  |  |  |
| --- | --- | --- |
| Руководитель дипломной работы |  | (подпись) |
| *преподаватель МРМТ Савинов Александр Николаевич* | | |
| (должность, место работы, ФИО) | | |

|  |  |  |
| --- | --- | --- |
| Задание принял к исполнению |  | Е.П. Валиев |
|  |  | (подпись, дата, ФИО) |

Задание рассмотрено на заседании цикловой комиссии информационных и радиотехнических дисциплин «\_\_\_»\_\_\_\_\_\_20\_\_\_г.

|  |  |  |
| --- | --- | --- |
| Председатель ЦК |  | Е.А.Муравьева |
| (подпись, ФИО) | | |

ОГЛАВЛЕНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 4](#_Toc169156355)

[ВВЕДЕНИЕ 5](#_Toc169156356)

[1 Теоретический раздел 7](#_Toc169156357)

[1.1 Понятие программного обеспечения 7](#_Toc169156358)

[1.2 Понятие базы данных 10](#_Toc169156359)

[2 Аналитический раздел 13](#_Toc169156360)

[2.1 Создание диаграммы прецедентов 13](#_Toc169156361)

[2.2 Выбор средств разработки информационной системы 17](#_Toc169156362)

[2.2.1 Выбор СУБД 17](#_Toc169156363)

[2.2.2 Выбор среды разработки и языка программирования 21](#_Toc169156364)

[3 Исследовательский раздел 27](#_Toc169156365)

[3.1 Техническое задание 27](#_Toc169156366)

[3.1.1 Назначение программы 27](#_Toc169156367)

[3.1.2 Интерфейс программы 27](#_Toc169156368)

[3.1.3 Требования к программе 28](#_Toc169156369)

[3.2 Разработка инфологической модели БД 28](#_Toc169156370)

[2.3 Разработка даталогической модели БД 30](#_Toc169156371)

[3.2 Разработка базы данных 33](#_Toc169156372)

[3.3 Разработка интерфейса пользователя 35](#_Toc169156373)

[3.5 Тестирование программного обеспечения 43](#_Toc169156374)

[ЗАКЛЮЧЕНИЕ 48](#_Toc169156375)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 49](#_Toc169156376)

[ПРИЛОЖЕНИЕ 50](#_Toc169156377)

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

GIT — Global Information Tracker (Глобальный информационный трекер);

SQL — Structured Query Language (Язык структурированных запросов);

UML — Unified Modeling Language (Унифицированный язык моделирования);

БД — база данных;

ПО — программное обеспечение;

СУБД — Система управления базами данных.

ВВЕДЕНИЕ

Современные учебные заведения нуждаются в современных технологиях. Почти каждый процесс максимально автоматизируется для простоты удобства пользования конечным пользователем, поэтому с самого начала необходимо заранее позаботится о новом учебном оборудовании учреждения.

Текущие системы учета основываются на ручном учете и часто сопровождаются ошибками, недопониманием и задержками. Это может привести к потере потенциальных студентов и снижению конкурентоспособности среди других учебных заведениях.

Для автоматизации и оптимизации процессов учета абитуриентов необходимо разработать специализированное программное обеспечение. Данная дипломная работа посвящена разработке программного продукта, который позволит любой технической школе вести учет абитуриентов, упрощая сопровождающие процессы обслуживания и анализа данных.

Целью данной дипломной работы является разработка программного обеспечения для учета абитуриентов техникума. Разработка такой системы позволит оптимизировать процессы учета ведения информации о абитуриентах, а также улучшить взаимодействие с потенциальными студентами.

Для достижения поставленной цели необходимо решить следующие задачи:

1. провести анализ предметной области и выбор средств разработки информационной системы;
2. спроектировать и разработать базу данных информационной системы;
3. разработать структуру программного обеспечения и алгоритм работы информационной системы;
4. разработать пользовательский интерфейс информационной системы;
5. провести тестирование программного обеспечения информационной системы.

Основной задачей разработки программного обеспечения для техникума является создание удобной и функциональной системы, способной автоматизировать процессы учета абитуриентов, обеспечить оперативный доступ к необходимой информации, а также повысить эффективность работы с базой данных.

В рамках данной работы будут рассмотрены основные этапы разработки программного обеспечения, проектирование архитектуры системы, выбор технологий и инструментов для реализации программного продукта, а также проведение тестирования и анализ результатов.

1 Теоретический раздел

* 1. Понятие программного обеспечения

Программное обеспечение — это набор программных инструкций, которые позволяют компьютеру выполнять определенные задачи. Оно играет ключевую роль в функционировании компьютерных систем, поскольку без программного обеспечения компьютер не сможет работать.

Важным аспектом ПО является его разнообразие и специализация. Оно может быть написано для выполнения конкретных задач, таких как обработка текста, графики, аудио и видео, управление базами данных, а также для обеспечения безопасности данных. Программное обеспечение также может быть классифицировано по лицензированию: проприетарная, свободная, с открытым исходным кодом и коммерческая лицензия.

Проприетарная лицензия — это лицензия, по которой издатель ПО даёт разрешение получателю использовать одну или несколько копий программы, но при этом сам остаётся правообладателем всех этих копий. Примеры продуктов с проприетарной лицензией включают в себя Microsoft Office, Adobe Photoshop, Oracle Database.

Открытая лицензия - ПО с открытым исходным кодом, которое может быть свободно изменено, распространено и использовано. Примеры включают Linux, Mozilla Firefox и VLC Media Player.

Бесплатная лицензия — это ПО, которое можно получить и использовать бесплатно, но может иметь ограничения в использовании или распространении. Примеры включают Avast Free Antivirus, Google Chrome и Audacity

Коммерческая лицензия — это лицензия, использующаяся для продажи ПО или предоставления доступа к нему на коммерческой основе. Примеры продуктов с коммерческой лицензией включают в себя Microsoft Windows, Adobe Creative Suite.

Программное обеспечение можно разделить на две основные категории: системное и прикладное. Системное программное обеспечение включает операционные системы, драйверы устройств и другие программы, обеспечивающие работоспособность компьютера. Оно отвечает за управление ресурсами компьютера и обеспечивает взаимодействие между аппаратным и программным обеспечением. Примерами системного программного обеспечения являются Windows, macOS и Linux.

Прикладное программное обеспечение, напротив, предназначено для выполнения определенных задач пользователем. Оно включает в себя программы для работы с текстом, изображениями, аудио и видео, а также специализированные программы для решения определенных задач.

Программное обеспечение создается программистами, использующими различные языки программирования, такие как Java, C++, Python и другие. Они разрабатывают программу, которая решает конкретную задачу и предоставляет пользователю интерфейс для взаимодействия с ней.

Программное обеспечение имеет свои плюсы и минусы. Среди преимуществ можно выделить следующие:

1. автоматизация задач: Программное обеспечение позволяет автоматизировать многие задачи, что повышает производительность и эффективность работы;
2. Разнообразие функций: существует большой выбор программ для различных нужд — от офисных приложений до специализированных программ для проектирования, программирования и других областей;
3. Повышение производительности - с помощью программного обеспечения вы можете автоматизировать и ускорить выполнение рутинных задач;
4. расширенные возможности - с помощью программного обеспечения вы можете создавать новые продукты, разрабатывать и оптимизировать бизнес-процессы;
5. Улучшение коммуникации: Многие приложения упрощают общение и совместную работу, облегчая обмен информацией между пользователями;
6. Экономия времени и ресурсов - Программное обеспечение может помочь сэкономить время и ресурсы при выполнении различных задач;
7. Развлечения и обучение: Программное обеспечение также используется в образовательных и развлекательных целях, что способствует личностному развитию и отдыху.

Недостатки программного обеспечения:

1. зависимость от производителей - при возникновении проблем с программным обеспечением может потребоваться помощь специалиста, что также связано с дополнительными расходами;
2. Безопасность - некачественное программное обеспечение или уязвимости могут стать источником угроз для пользовательских данных, что требует постоянного внимания к обновлениям и безопасности;
3. Совместимость - не все программы совместимы друг с другом или с разными операционными системами, что может вызвать проблемы при работе с разными форматами файлов или сетевыми условиями.;
4. Высокая стоимость - разработка высококачественного программного обеспечения может быть дорогостоящей;
5. необходимость в постоянных обновлениях - для поддержания функциональности и безопасности приложения требуются постоянные обновления и поддержка;
6. риск возникновения ошибок и сбоев - некорректная работа программного обеспечения может привести к серьезным проблемам и потере данных

Программное обеспечение играет ключевую роль в современном обществе, предоставляя широкие возможности для повышения производительности и расширения возможностей пользователей, однако требует внимательного подхода к выбору и использованию.

* 1. Понятие базы данных

База данных (БД) – это организованная коллекция данных, которая обычно хранится и обрабатывается с использованием специализированного программного обеспечения. Она представляет собой централизованное хранилище информации о предметной области, которая может быть легко доступна и управляема.

Основное назначение БД – это обеспечить доступ к информации в удобной и эффективной форме. Они помогают хранить огромные объемы данных, структурировать их, обеспечивать целостность и безопасность информации, а также управлять доступом к данным различным пользователям.

Базы данных используются в различных областях, включая бизнес, образование, науку, здравоохранение и многие другие. Они позволяют организациям хранить и управлять огромными объемами информации, делая её доступной для анализа, принятия решений и автоматизации бизнес-процессов.

Классификация БД изображена на рисунке 1.

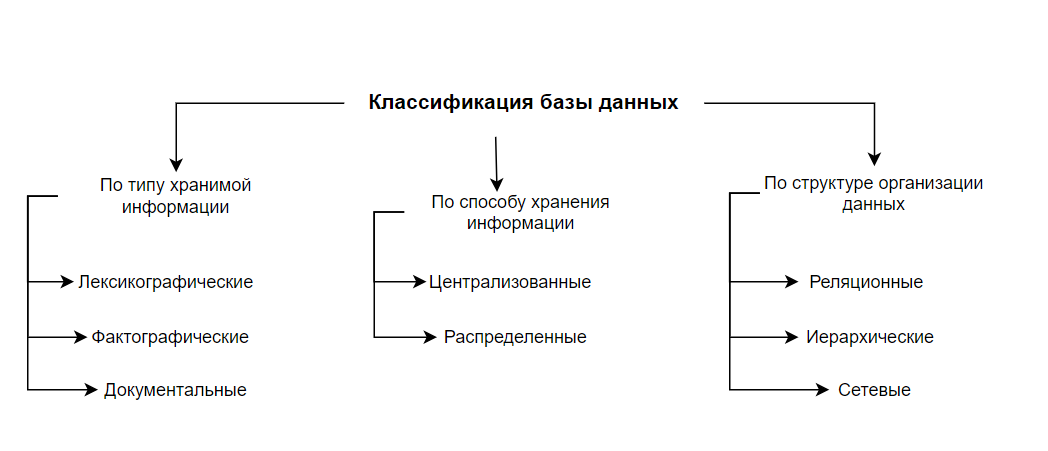


Рисунок 1 — классификация базы данных

Фактографические БД – это БД, которые содержат информацию, относящуюся непосредственно к предметной области. Их в основном используют в экономике и управлении.

Лексикографические БД – это разнообразные словари, глоссарии, кодификаторы, своды, рубрикаторы, которые применяются как справочники вместе с фактографическими БД.

Документальные БД – это БД, хранящие документы, сгруппированные по конкретным, заданным особенностям.

Интегрированные или централизованные БД дают право на доступ к информации определенной группе пользователей.

Распределительные БД полностью копируют интегрированные БД, но в отличии от них имеют возможность находиться на разных устройствах физически, но считаться одним целым с точки зрения логики.

Реляционная база данных – база данных, основанная на связях, отношениях и зависимостях. Самая популярная и самая приближенная к алгебре модель баз данных. Примеры: MySQL, PostgreSQL, Oracle.

Иерархическая модель данных – все данные представлены с помощью иерархической структуры, «родителей» и «наследников», данные распределены по разным уровням. Пример: XML.

Сетевая модель данных – основаны на модели сети, где данные представлены в виде графа. Каждая запись может иметь несколько родителей и дочерних элементов. Пример: IDMS.

Основные компоненты базы данных включают:

1. Структура данных — это описание данных и их отношений друг с другом. Она определяет, как данные организованы и как они связаны между собой.

2. Язык запросов — это набор команд, используемых для взаимодействия с базой данных. Язык запросов позволяет пользователю извлекать, обновлять и управлять данными в базе.

3. Система управления базами данных (СУБД) — это программное обеспечение, которое обеспечивает создание, управление и доступ к базе данных. Оно предоставляет интерфейс для работы с данными, выполнения запросов и обеспечивает безопасность и целостность данных.

Базы данных могут быть использованы для решения различных задач, включая:

* хранение данных: БД позволяют эффективно хранить большие объемы информации, такие как информация о клиентах, продукции, заказах и т.д.;
* управление данными: БД обеспечивают механизмы для добавления, обновления и удаления данных, поддерживая их целостность и безопасность;
* поиск и извлечение данных: с помощью языка запросов, такого как SQL, пользователи могут выполнять сложные запросы для поиска и извлечения нужной информации из базы данных;
* обеспечение безопасности: СУБД позволяют устанавливать права доступа к данным, шифровать конфиденциальную информацию и реализовывать механизмы аутентификации пользователей.
* резервное копирование и восстановление: БД предоставляют средства для создания резервных копий данных и восстановления информации в случае сбоев или потерь.

2 Аналитический раздел

2.1 Создание диаграммы прецедентов

Процесс создания программного обеспечения начинается с этапа анализа предметной области и решаемых в ней задач. Для упрощения процесса анализа решено создать и использовать набор диаграмм согласно нотации UML.

Язык UML представляет собой общецелевой язык визуального моделирования, разработанный для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем. Этот язык одновременно является простым и мощным средством моделирования и может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

Язык UML предназначен для решения следующих основных задач:

* Предоставить в распоряжение пользователей легко воспринимаемый и выразительный язык визуального моделирования, специально предназначенный для разработки и документирования моделей сложных систем самого различного целевого назначения.
* Снабдить исходные понятия языка UML возможностью расширения и специализации для более точного представления моделей систем в конкретной предметной области
* Поддерживать такую спецификацию моделей, которая не зависит от конкретных языков программирования и инструментальных средств проектирования программных систем

В языке UML используется четыре основных вида графических конструкций:

* значки, или пиктограммы;
* графические символы на плоскости;
* пути;
* строки текста.

При графическом изображении диаграмм следует придерживаться следующих основных рекомендаций:

* каждая диаграмма должна служить законченным представлением;
* соответствующего фрагмента моделируемой предметной области;
* вся информация о сущностях должна быть явно представлена на диаграммах;
* диаграммы не должны содержать противоречивой информации;
* диаграммы не следует перегружать текстовой информацией.

Визуальное моделирование в UML можно представить, как некоторый процесс поуровневого спуска от наиболее общей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме так называемой диаграммы прецедентов (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования.

Суть данной диаграммы состоит в том, что проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых прецедентов.

Актеры – это роли, выполняемые сущностями, внешними по отно­шению к системе, которые взаимодействуют непосредственно с сис­темой. Выявить актеров можно, выяснив, кто или что использует, или взаимодействует непосредственно с системой.

По результатам анализа предметной области была построена диаграмма прецедентов, включающая в себя следующие элементы:

1. набор актеров (пользователей разрабатываемой программы):

* Администратор – добавление, удаление, редактирование и просмотр специальностей техникума;
* Регистратор – просмотр, поиск, добавление, удаление и редактирование личной информации абитуриента.

1. основные прецеденты программы:

* просмотр, добавление и редактирование специальности – администратор добавляет и редактирует специальность в базе данных;
* просмотр меню, добавление, удаление и редактирование абитуриента – регистратор редактирует информацию в базе данных.

Процесс построения диаграммы прецедентов включал в себя следующие этапы:

* создание блоков для каждого прецедента;
* указание основных актеров, взаимодействующих с каждым прецедентом;
* установление связей между прецедентами, отображающих порядок выполнения действий.

Используя все выше написанные данные, создана диаграмма прецедентов, изображенная на рисунке 2.

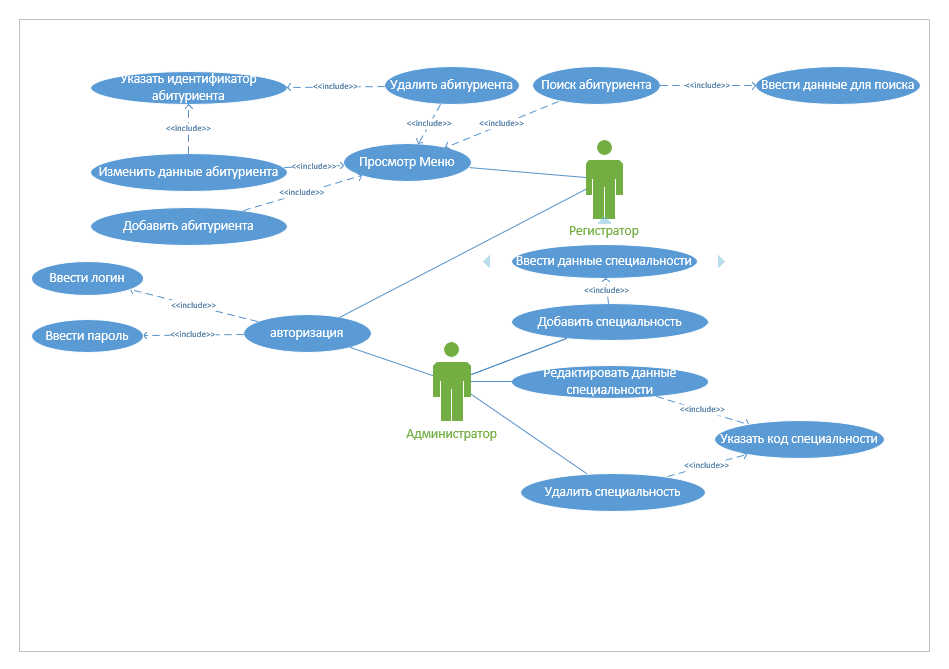


Рисунок 2 – Диаграмма прецедентов

2.2 Выбор средств разработки информационной системы

2.2.1 Выбор СУБД

Разрабатываемая программа основана на хранении данных в базе данных. В связи с этим необходимо выбрать СУБД, в которой будет создана база данных.

Система управления базами данных (СУБД) - это тип программного обеспечения, которое взаимодействует с самой базой данных, приложениями и пользовательскими интерфейсами для получения данных и их анализа. СУБД также содержит ключевые инструменты для управления данными.

Популярные и основные реляционные базы данных

* SQLite — очень мощная встраиваемая система управления;
* MySQL — самая популярная и распространённая СУБД;
* PostgreSQL — наиболее продвинутая СУБД.

Рассмотрим каждый из них:

1. MySQL (My Structured Query Language) - то система управления реляционными базами данных, которая позволяет хранить и управлять данными. Она является одной из самых популярных и широко используемых СУБД в мире. Она предлагает надежное и эффективное хранение данных, отличается высокой производительностью, имеет богатый функционал и поддерживает множество типов данных. MySQL также предоставляет широкие возможности для администрирования баз данных и обеспечивает высокую степень безопасности данных.

Выделены основные преимущества MySQL:

* бесплатный доступ;
* предоставляет большой функционал;
* безопасность;
* простота использования: MySQL легок для изучения и использования.
* подходит для больших баз данных;
* множественный контроль доступа;
* скорость;
* широкая поддержка: MySQL поддерживается множеством платформ и операционных систем, что обеспечивает удобство его использования.

Выделены основные недостатки MySQL:

* ограниченная функциональность: MySQL может не поддерживать некоторые более продвинутые функции и возможности, которые предлагают другие СУБД;
* слабая поддержка для более продвинутой аналитики и обработки данных: в некоторых случаях может понадобиться дополнительное программное обеспечение или расширения для работы с данными в MySQL.

Задачи, в которых MySQL является лучшим вариантом:

* многопользовательский доступ к приложениям;
* когда требуются надежность безопасности и аутентификации;
* с распределенными системами;
* приложения с большим объемом данных;
* проекты, которые требуют большой масштабируемости;
* веб-приложения;
* при разработке индивидуальных решений.

MySQL намного более гибок в отношении типов данных по сравнению с SQLite.

В целом, MySQL является популярным и широко используемым выбором для многих проектов. Он обладает хорошей производительностью, надежностью и доступностью, но иногда может иметь ограничения в функциональности и масштабируемости по сравнению с другими более продвинутыми СУБД.

1. PostgreSQL — это мощная и расширяемая СУБД с открытым исходным кодом. PostgreSQL обладает хорошей поддержкой стандартов SQL, простотой в использовании и обширным набором функций. Однако его производительность может быть несколько хуже, чем у MySQL, особенно при работе с большими объемами данных.

Выделены основные преимущества PostgreSQL:

* открытый исходный код;
* расширенные возможности: поддержка множества расширенных функций, таких как географические объекты, JSON, XML, полнотекстовый поиск и другие;
* безопасность;
* масштабируемость;
* активное сообщество: вокруг PostgreSQL сформировалось большое сообщество разработчиков и пользователей, которые предоставляют поддержку, создают расширения и улучшают систему.

Выделены основные недостатки PostgreSQL:

* сложность конфигурации;
* меньшее распространение;
* производительность;
* отсутствие интеграции с некоторыми инструментами: несмотря на хорошую интеграцию PostgreSQL с многими языками программирования и фреймворками, некоторые инструменты и приложения могут иметь ограниченную поддержку или требовать дополнительных настроек;
* большое потребление ресурсов для работы.

1. SQLite, в свою очередь, является более легкой и компактной СУБД, которая хорошо подходит для небольших проектов или для мобильной разработки. Однако, SQLite имеет ограниченные возможности для работы с большими объемами данных и не всегда обеспечивает высокую производительность.

Выделены основные преимущества SQLite:

1. Легковесность: SQLite является небольшим, самодостаточным и легковесным движком базы данных, который не требует установки сервера или сложной конфигурации.
2. Простота использования;
3. Поддержка множества платформ.
4. Высокая производительность при работе с небольшими объемами данными.
5. Надежность.

Выделены основные недостатки SQLite:

1. Низкая масштабируемость: SQLite не предназначен для работы с большими объемами данных или высоконагруженными проектами, и может проявиться недостаточная производительность при больших запросах.
2. Ограниченная функциональность.
3. Отсутствие многопользовательской поддержки: SQLite не поддерживает одновременный доступ к базе данных для нескольких пользователей, что может быть ограничением для некоторых приложений и проектов.
4. Отсутствие некоторых типов данных и операций: Некоторые продвинутые функции и типы данных, такие как полнотекстовый поиск или географические объекты, могут быть ограничены или недоступны в SQLite.

В целом, SQLite может быть отличным выбором для небольших проектов, прототипов, мобильных приложений или встроенных систем, но возможны ограничения при работе с большими объемами данных или сложными приложениями.

Для упрощения выбора была создана таблица 1, на которой сравниваются вышеперечисленные СУБД.

Таблица 1 – Сравнение СУБД.

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | MySQL | SQLite | PostgreSQL |
| Тип | Реляционная СУБД | Встроенная реляционная СУБД | Расширяемая реляционная СУБД |
| Лицензия | Бесплатная, открытый исходный код | Домен общественного пользования | BSD-style |
| Масштабируемость | Средняя | Низкая | Высокая |
| Производительность | Высокая | Ограниченная | Высокая |
| Сложность настройки | Средняя | Низкая | Высокая |
| Поддержка SQL | Полная | Ограниченная | Полная |
| Многопоточность | Поддерживается | Не поддерживается | Поддерживается |
| Типы данных | Поддерживает широкий спектр | Ограниченный спектр | Поддерживает широкий спектр |
| Сложные запросы | Поддерживает | Ограниченная поддержка | Полная поддержка |

По результатам сравнения СУБД, сделан вывод в пользу MySQL. Он обладает отличной производительностью, широким набором функций и инструментов, а также удобным языком запросов SQL. MySQL подходит как для небольших проектов, так и для крупных приложений с большим объемом данных.

2.2.2 Выбор среды разработки и языка программирования

Согласно требованиям, необходимо было создать настольное Windows-приложение с графическим интерфейсом пользователя. Рассмотрим самые популярные среды разработки, подходящие под требования.

Visual Studio и C# являются популярными инструментами для разработки приложений под платформу .NET. Они предоставляют богатую интегрированную среду разработки (IDE) и мощный язык программирования, которые позволяют разработчикам быстро и эффективно создавать высококачественное программное обеспечение.

Выделим главные преимущества Visual Studio:

* одним из главных преимуществ VS является его интеграция с различными языками программирования, такими как C++, C# и другие. Это позволяет программистам выбирать наиболее подходящий язык на основе их потребностей, что приводит к более легкому и эффективному написанию кода.;
* обладает удобным и понятным интерфейсом, который делает его простым в использовании для новых пользователей, а также для опытных разработчиков. Этот инструмент также поддерживает широкий спектр функций отладки и тестирования, позволяя быстро выявлять ошибки и устранять их;
* интеграция с Git — Это упрощает совместную работу над большими проектами с другими разработчиками;
* расширяемость - имеет множество доступных плагинов и дополнений;
* инструменты отладки - Встроена мощная система отладки, позволяющая быстро находить и исправлять ошибки в коде;
* удобная редакция кода - Визуальный интерфейс Visual Studio обладает удобным и гибким редактором кода, включающим различные функции сортировки и подсветки синтаксиса;
* объединение проектов - В VS 2019 можно объединить проекты разных типов: например, можно объединить проект, написанный на C++, с проектом Microsoft Access.

Visual Studio также обладает мощным сообществом программистов, которые предлагают множество ресурсов и инструментов для обучения и поддержки. Это делает Visual Studio более доступным для широкого круга пользователей.

Одним из конкурентов Visual Studio является среда разработки JetBrains Rider. Она предлагает более легкий и быстрый интерфейс, который может быть удобнее для определенных пользователей. Кроме того, имеет некоторые уникальные функции, которых нет в Visual Studio.

Еще одним примером среды разработки является SharpDevelop. Она предлагает открытый исходный код и бесплатную лицензию, что может быть привлекательным для разработчиков с ограниченным бюджетом. Однако, SharpDevelop имеет меньше функций и инструментов, чем Visual Studio.

Еще одно сравнение - с редактором Visual Studio Code (VS Code). Несмотря на то, что VS Code является легким и легко настраиваемым, в нем отсутствуют некоторые расширенные функции и интеграции, присущие Visual Studio. Однако для небольших проектов или разработчиков, предпочитающих минималистский подход, VS Code может стать отличным решением.

Результаты сравнительного анализа оформлены и представлены в таблице 2.

Таблица 2 – Сравнение сред разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Особенность | Visual Studio | SharpDevelop | JetBrains Rider |
| Поддержка языков | C#, C++, Visual Basic | C#, Visual Basic, F#, Boo | C#, VB.NET, F#, ASP.NET, SQL |
| Интеграция | Широкий выбор интеграций с различными платформами и сервисами | Ограниченные возможности интеграции | Интеграция с платформами JetBrains, компилятор Roslyn |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| Особенность | Visual Studio | SharpDevelop | JetBrains Rider |
| Расширяемость | Большое количество доступных плагинов и расширений | Ограниченная поддержка расширений | Поддержка большого количества плагинов и расширений |
| Оперативная память | Требует много ресурсов | Более легковесный по сравнению с Visual Studio | Требователен к ресурсам компьютера |
| Скорость работы | Может быть медленным на старых компьютерах | Более быстрый, чем Visual Studio | Быстрая работа, оптимизирована для эффективной разработки |
| Стоимость | Платная версия и бесплатная версия для студентов и некоммерческого использования | Бесплатное ПО | Платное ПО с пробным периодом |

Итак, Visual Studio обладает широким набором функций и интеграций, но требует много ресурсов компьютера. SharpDevelop является более легковесным вариантом, но ограничен в возможностях расширения. JetBrains Rider предлагает широкий набор инструментов для разработки, но может быть требователен к ресурсам компьютера.

Сравнивая среды разработки, было решено выбрать Visual Studio, так как VS обладает богатым набором инструментов и функций, которые делают разработку программного обеспечения более удобной и эффективной.

В качестве используемого языка программирования, выбран язык C#. Он был создан в 2000 году и является частью платформы .NET Framework. C# сочетает в себе простоту, эффективность и безопасность, делая его популярным выбором для разработки различных приложений, включая веб-приложения, приложения для Windows, мобильные приложения и игры.

Некоторые ключевые особенности C#:

1. сильная типизация: все переменные должны быть объявлены с указанием их типа данных;
2. управляемый язык: С# использует сборку мусора для автоматического освобождения памяти;
3. LINQ: позволяет писать SQL-подобные запросы к структурам данных прямо в коде C#;
4. обширная стандартная библиотека: C# поставляется с широким набором классов и методов, упрощающих разработку приложений.

Причины, по которым был выбран язык C# для разработки ПО:

1. универсальность: C# является универсальным языком программирования, который используется для создания различных видов приложений, включая веб-сайты, настольные приложения, мобильные приложения и игры;
2. интеграция с платформой .NET: C# является основным языком программирования для платформы .NET, что обеспечивает широкие возможности для разработки и интеграции приложений;
3. богатые возможности: C# обладает богатым набором функциональных возможностей, включая объектно-ориентированное программирование, обработку событий, многопоточность и т.д.;
4. легкость для изучения и использования: C# имеет четкий и понятный синтаксис что делает его относительно легким для изучения и использования, особенно для студентов;
5. поддержка современных технологий: C# активно развивается и поддерживает современные технологии, такие как облачные вычисления, искусственный интеллект, машинное обучение и т.д.;
6. широкая поддержка и сообщество: C# имеет широкую международную поддержку и активное сообщество разработчиков, что обеспечивает доступ к обучающим материалам, библиотекам и инструментам для разработки;

Таким образом, выбор языка программирования C# для дипломного проекта обусловлен его удобством, широкими возможностями интеграции с базами данных, мощными и удобными инструментами для разработки программного обеспечения.

3 Исследовательский раздел

3.1 Техническое задание

3.1.1 Назначение программы

Целью разрабатываемой программы является создание удобного и эффективного настольного приложения для учёта абитуриентов техникума. Программа помогает оптимизировать рабочие процессы, увеличивает эффективность работы персонала.

С помощью разрабатываемого программного обеспечения можно автоматизировать различные задачи, такие как:

1. управление данными о абитуриентах: создание, редактирование, удаление и хранение информации;
2. Управление специальностями: создание, редактирование, удаление и хранение информации.

В целом, назначение программы заключается в автоматизации рутинных процессов, оптимизации использования ресурсов и предоставления актуальной информации для принятия решений.

3.1.2 Интерфейс программы

В процессе разработки программы будет создано:

1. главное окно программы, которое позволяет выбрать необходимые функции:

* авторизация пользователя;
* сохранение логина и пароля;
* управление данными о подключении к серверу.

1. окно администратора, которое позволяет выбрать необходимые функции:

* добавить специальность;
* изменение данных специальности;
* удаление специальности.

1. окно регистратора, которое позволяет выбрать необходимые функции:

* добавить абитуриента;
* изменение данных абитуриента;
* удаление абитуриента;
* поиск абитуриента.

3.1.3 Требования к программе

Основные требования для работы программы:

* устойчива к ошибкам пользователя, должна проводить валидацию вводимых данных и выводить сообщения об ошибках при неправильном вводе;
* интуитивно понятна и удобна в использовании;
* интеграция с базой данных для хранения и обработки информации.

3.2 Разработка инфологической модели БД

Инфологическая модель базы данных разрабатывается с целью определения структуры данных, их взаимосвязей и организации хранения информации. Это позволяет проектировать эффективную структуру базы данных, которая будет соответствовать требованиям и задачам исследования.

Для информационной системы учёта абитуриентов выделены следующие сущности:

* пользователи: сущность содержит логин и пароль, а также тип пользователя;
* абитуриент: сущность содержит идентификатор, а также Ф.И.О. абитуриента;
* Специальность: сущность содержит код, полное и краткое наименования специальности;
* Персональные данные абитуриентов: сущность содержит идентификатор персональных данных, идентификатор абитуриента, общий бал за экзамены, сведения об инвалидности, льготах и статусе сироты, а также комментарии к документам и абитуриенту;
* Специальности абитуриентов: сущность содержит идентификаторы операции, абитуриента и специальности;
* Экзамены абитуриентов: сущность содержит идентификатор операции, абитуриента, а также баллы полученные абитуриентом за прохождение экзаменов;
* Экзамены специальностей: сущность содержит идентификаторы операции и кода специальности, а также 4 необходимых экзамена для специальности.

Разработанная инфологическая модель базы данных изображена на рисунке 3.

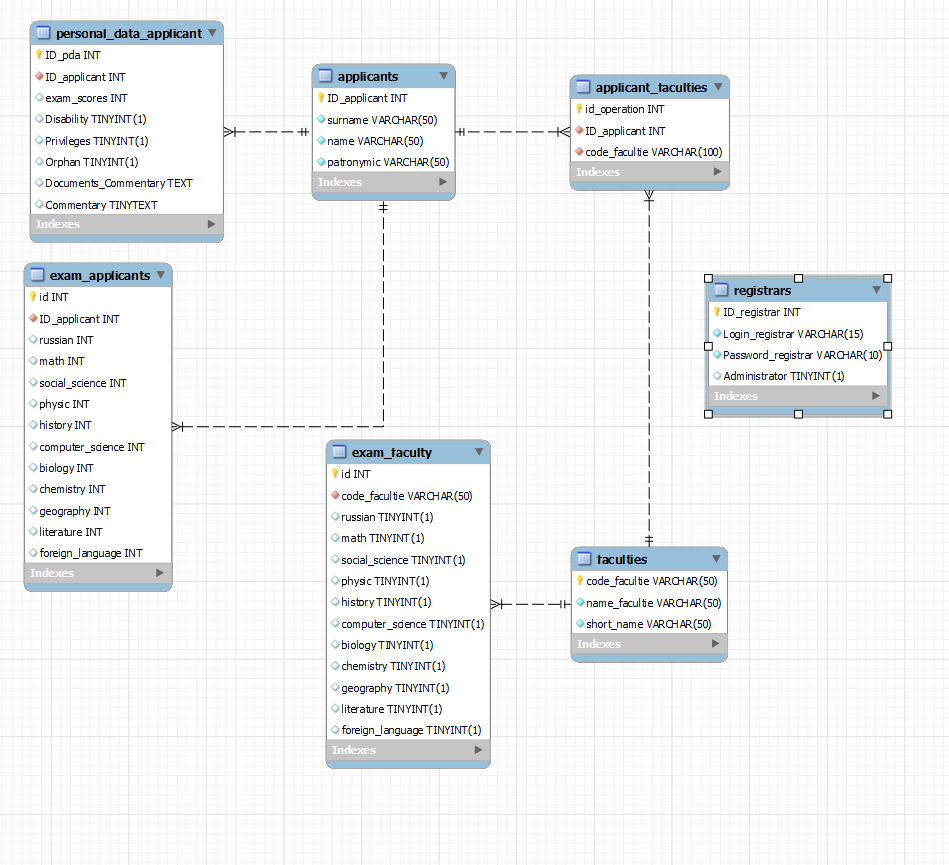


Рисунок 3 – инфологическая модель БД

2.3 Разработка даталогической модели БД

Для перехода от инфологической модели к реляционной существует алгоритм:

1. каждой сущности ставится в соответствие отношение;
2. каждому атрибуту сущности ставится в соответствие соответствующий атрибут соответствующего отношения;
3. первичные ключи для каждой из сущностей. Рекомендуется использовать суррогатные первичные ключи (идентификаторы) вместо реальных ключевых атрибутов;
4. определить внешние ключи для каждого из отношений на основе анализа кардинальности каждого из отношений.

Опишем каждую сущность инфологической модели. Данные представлены в Таблицах 3 - 9.

Таблица 3 – Данные сущности «Пользователи»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 1 | ID\_registrar | INT | 2147483647 | PK |
| 2 | Login\_registrar | VARCHAR | 15 | — |
| 3 | Password\_registrar | VARCHAR | 10 | — |
| 4 | Administrator | TINYINT | 1 | — |

Таблица 4- Данные сущности «абитуриент»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 1 | ID\_applicant | INT | 2147483647 | PK |
| 2 | surname | VARCHAR | 50 | — |
| 3 | name | VARCHAR | 50 | — |
| 4 | patronymic | VARCHAR | 50 | — |

Таблица 5 – Данные сущности «Специальность»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 1 | code\_facultie | VARCHAR | 50 | PK |
| 2 | name\_facultie | VARCHAR | 50 |  |
| 3 | short\_name | VARCHAR | 50 |  |

Таблица 6 – Данные сущности «Персональные данные абитуриентов»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 1 | ID\_pda | INT | 2147483647 | PK |

Продолжение таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 2 | ID\_applicant | INT | 2147483647 | FK |
| 3 | exam\_scores | INT | 2147483647 | — |
| 4 | Disability | TINYINT | 1 | — |
| 5 | Privileges | TINYINT | 1 | — |
| 6 | Orphan | TINYINT | 1 | — |
| 7 | Documents\_Commentary | TEXT | 65535 | — |
| 8 | Commentary | TINYTEXT | 255 | — |

Таблица 7 – Данные сущности «Специальности абитуриентов»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 1 | id\_operation | INT | 2147483647 | PK |
| 2 | ID\_applicant | INT | 2147483647 | FK |
| 3 | code\_facultie | VARCHAR | 100 | FK |

Таблица 8 – Данные сущности «Экзамены абитуриентов»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 1 | id | INT | 2147483647 | PK |
| 2 | ID\_applicant | INT | 2147483647 | FK |
| 3 | russian | INT | 2147483647 | — |
| 4 | math | INT | 2147483647 | — |
| 5 | social\_science | INT | 2147483647 | — |
| 6 | physic | INT | 2147483647 | — |
| 7 | history | INT | 2147483647 | — |
| 8 | computer\_science | INT | 2147483647 | — |
| 9 | biology | INT | 2147483647 | — |
| 10 | chemistry | INT | 2147483647 | — |
| 11 | geography | INT | 2147483647 | — |
| 12 | literature | INT | 2147483647 | — |
| 13 | foreign\_language | INT | 2147483647 | — |

Таблица 9 – Данные сущности «Экзамены специальностей»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 1 | id | INT | 2147483647 | PK |
| 2 | code\_facultie | VARCHAR | 50 | FK |
| 3 | russian | TINYINT | 1 | — |
| 4 | math | TINYINT | 1 | — |
| 5 | social\_science | TINYINT | 1 | — |
| 6 | physic | TINYINT | 1 | — |
| 7 | history | TINYINT | 1 | — |
| 8 | computer\_science | TINYINT | 1 | — |
| 9 | biology | TINYINT | 1 | — |

Продолжение таблицы 9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Названия поля | Тип данных | Размер | Ключ |
| 10 | chemistry | TINYINT | 1 | — |
| 11 | geography | TINYINT | 1 | — |
| 12 | literature | TINYINT | 1 | — |
| 13 | foreign\_language | TINYINT | 1 | — |

3.2 Разработка базы данных

Для создания базы данных используется конструкция CREATE DATABASE, которая имеется следующий обобщенный формат CREATE DATABASE имя\_базы\_данных;

Для описания новой таблицы, ее полей и индексов используется конструкция CREATE TABLE, которая имеет следующий обобщенный формат CREATE TABLE <имя таблицы>.

Для описания новой процедуры, ее аргументов используется конструкция CREATE PROCEDURE, которая имеет следующий обобщенный формат CREATE PROCEDURE <имя процедуры> (<аргумент 1> тип данных, <аргумент 2> тип данных, …, <аргумент n> тип данных).

Для описания новой функции, ее аргументов используется конструкция CREATE FUNCTION, которая имеет следующий обобщенный формат CREATE FUNCTION <имя функции> (<аргумент 1> тип данных, <аргумент 2> тип данных, …, <аргумент n> тип данных). В отличие от процедуры функция должна возвращать значение и иметь следующий синтаксис RETURNS <тип данных>.

Для информационной системы согласно даталогической модели БД была разработана база данных в СУБД MySQL, показанная на рисунке 4.

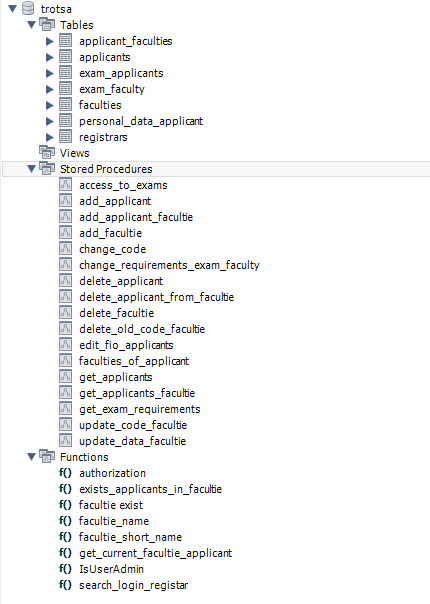


Рисунок 4 — база данных

3.3 Разработка интерфейса пользователя

Для разработки интерфейса пользователя решено использовать инструменты Windows Forms. Этот инструмент позволяет создавать графические интерфейсы приложений, поддерживает множество виджетов и элементов управления, а также обеспечивает легкую интеграцию с другими языками программирования.

Приложение состоит 6 форм:

1. Авторизация

Форма предназначена для авторизации пользователя с возможностью сохранения логина и пароля, а также для ввода данных сервера с возможностью проверить подключение. Форма показана на рисунках 5-6.

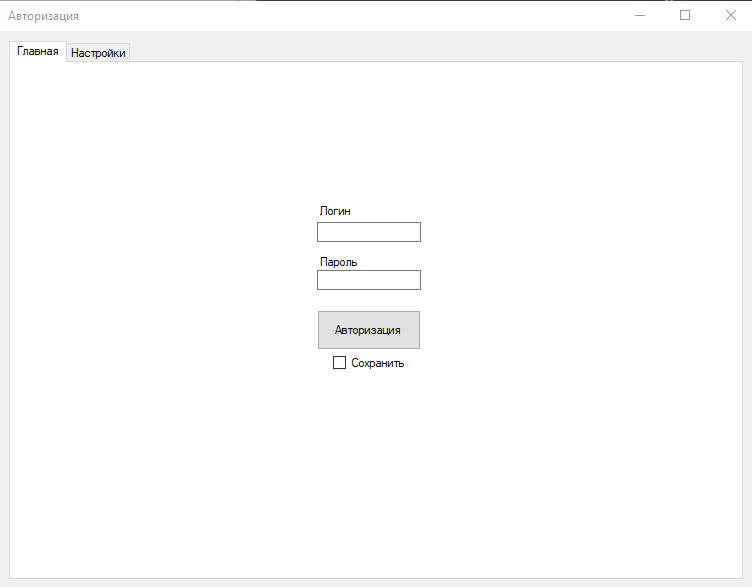


Рисунок 5 — форма авторизации, панель авторизации

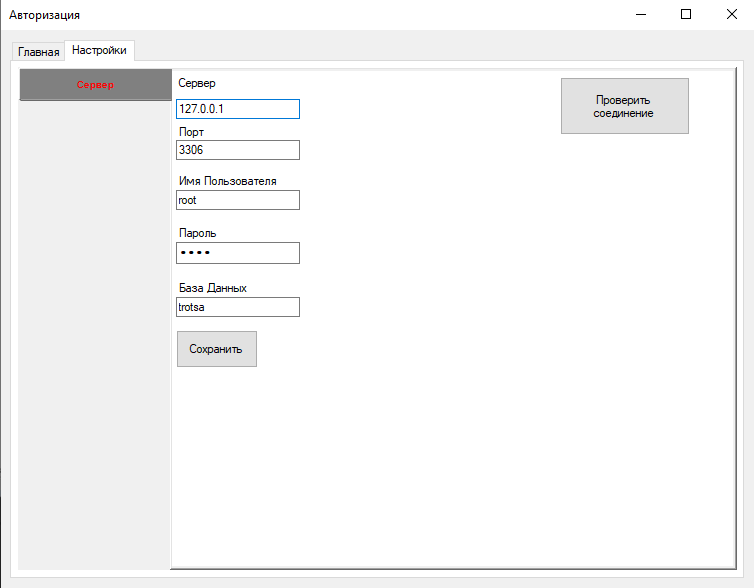


Рисунок 6 — форма авторизации, панель настроек

1. Администратор

Форма предназначена для администрирования специальностей и требующих для них экзаменов. Форма показана на рисунке 7.

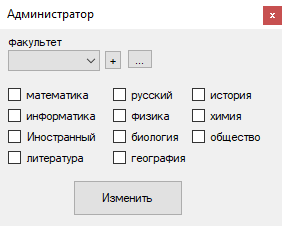


Рисунок 7 — форма администратора

1. специальность

Форма предназначена для добавления, удаления и редактирования специальности, которая будет использоваться в дальнейшей работе регистраторов с абитуриентами. Ответственность за специальности несёт администратор Форма показана на рисунках 8-9.

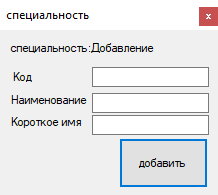


Рисунок 8 — форма специальности, добавление

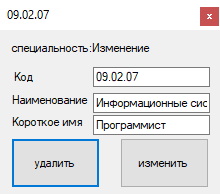


Рисунок 9 — форма специальности, редактирование

1. Регистратор

Форма предназначена для поиска абитуриентов. Форма показана на рисунке 10.

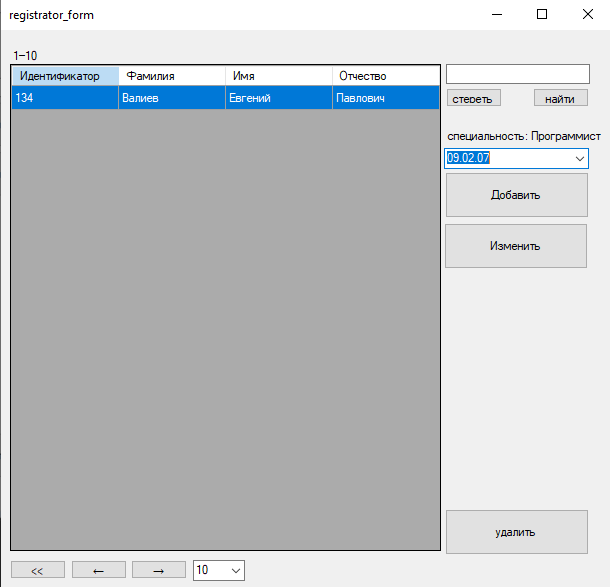


Рисунок 10 — форма регистратора

1. Редактирование абитуриента

Форма предназначена для редактирования личной информации абитуриента. Форма показана на рисунке 11.

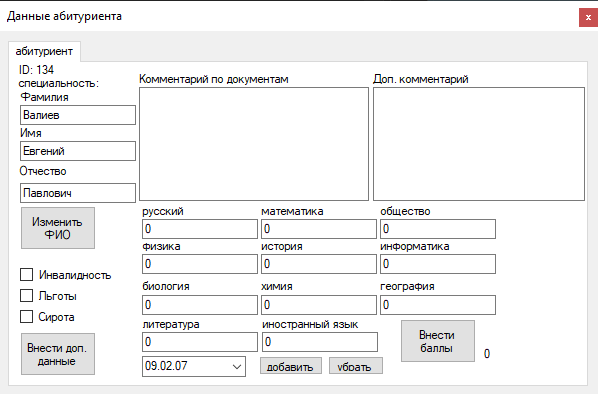


Рисунок 11 — форма редактирования личной информации абитуриента

1. Добавление абитуриента

Форма предназначена для добавления нового абитуриента. Форма показана на рисунке 12.

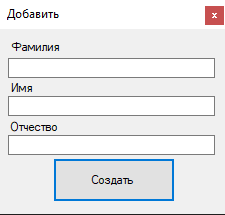


Рисунок 12 — форма добавления нового абитуриента

На формах присутствуют такие элементы как:

1. Label
2. Textbox
3. Button
4. TabPage
5. CheckBox
6. ComboBox
7. DataGridView

В программе использованы следующие фрагменты кода:

1. Авторизация:

private void button\_authorization\_Click(object sender, EventArgs e)

{

errorProvider\_MF.Clear();

if (registar.bun\_date > DateTime.Now)

{

if(authorization\_err\_count != 0)

{

MessageBox.Show($"Неудачных попыток авторизации: {authorization\_err\_count}\nПовторная попытка может быть совершена {registar.bun\_date}", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show($"Было совершено много неудачных попыток авторизации\nПовторная попытка может быть совершена {registar.bun\_date}", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

return;

}

if (login\_EditLine\_MF.Text == "")

{

errorProvider\_MF.SetError(login\_EditLine\_MF, "Введите логин");

return;

}

if(pass\_EditLine\_MF.Text == "")

{

errorProvider\_MF.SetError(pass\_EditLine\_MF, "Введите пароль");

return;

}

if (!db\_obj.search\_login(login\_EditLine\_MF.Text))

{

errorProvider\_MF.SetError(login\_EditLine\_MF, "Такого логина не существует");

}

if(db\_obj.authorization(login\_EditLine\_MF.Text, pass\_EditLine\_MF.Text))

{

if(!check\_save\_registar.Checked)

{

files\_work.clear\_registar\_data\_file();

login\_EditLine\_MF.Text = "";

pass\_EditLine\_MF.Text = "";

}

else

{

registar.login = login\_EditLine\_MF.Text;

registar.password = pass\_EditLine\_MF.Text;

files\_work.save\_registar\_data(registar);

}

this.Hide();

if(db\_obj.IsUserAdmin(login\_EditLine\_MF.Text))

{

administratorForm Adform = new administratorForm(this);

Adform.Show();

}

else

{

registrator\_form registrator\_Form = new registrator\_form(this);

registrator\_Form.Show();

}

}

else

{

errorProvider\_MF.SetError(pass\_EditLine\_MF, "Неверный пароль");

authorization\_err\_count++;

}

bunning(authorization\_err\_count);

}

1. Добавление абитуриента

private void button\_add\_applicant\_Click(object sender, EventArgs e)

{

if(DialogResult.Yes == MessageBox.Show("Вы точно хотите добавить абитуриента?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question))

{

try

{

obj\_addApplicants\_db.add\_applicant(textBox\_surname\_edit.Text, textBox\_name\_edit.Text, textBox\_patronymic\_edit.Text);

MessageBox.Show("Абитуриент добавлен", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Неудача", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

1. Добавление или изменение специальности

private void button\_add\_change\_Click(object sender, EventArgs e)

{

if (textBox\_code\_facultie.Text == "" || textBox\_name\_facultie.Text == ""

|| textBox\_short\_name.Text == "")

{

MessageBox.Show("Заполните все данные!", "ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (this.Text != "специальность")

{

if(obj\_Facultie.id == textBox\_code\_facultie.Text && obj\_Facultie.name == textBox\_name\_facultie.Text

&& obj\_Facultie.shrt\_name == textBox\_short\_name.Text)

{

MessageBox.Show("Вы ничего не поменяли!", "Изменение", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if(obj\_Facultie.id == textBox\_code\_facultie.Text && (obj\_Facultie.name != textBox\_name\_facultie.Text

|| obj\_Facultie.shrt\_name != textBox\_short\_name.Text))

{

obj\_facultie\_db.change\_data\_facultie(textBox\_code\_facultie.Text, textBox\_name\_facultie.Text, textBox\_short\_name.Text);

}

else if(obj\_Facultie.id != textBox\_code\_facultie.Text && obj\_Facultie.name == textBox\_name\_facultie.Text

&& obj\_Facultie.shrt\_name == textBox\_short\_name.Text)

{

obj\_facultie\_db.change\_code\_facultie(obj\_Facultie.id, textBox\_code\_facultie.Text);

obj\_facultie\_db.delete\_old\_code\_facultie(obj\_Facultie.id);

}

else if(obj\_Facultie.id != textBox\_code\_facultie.Text && (obj\_Facultie.name != textBox\_name\_facultie.Text

|| obj\_Facultie.shrt\_name != textBox\_short\_name.Text))

{

obj\_facultie\_db.change\_data\_facultie(textBox\_code\_facultie.Text, textBox\_name\_facultie.Text, textBox\_short\_name.Text);

obj\_facultie\_db.change\_code\_facultie(obj\_Facultie.id, textBox\_code\_facultie.Text);

obj\_facultie\_db.delete\_old\_code\_facultie(obj\_Facultie.id);

}

obj\_Facultie.id = textBox\_code\_facultie.Text;

obj\_Facultie.name = textBox\_name\_facultie.Text;

obj\_Facultie.shrt\_name = textBox\_short\_name.Text;

this.Text = obj\_Facultie.id;

}

else

{

if (obj\_facultie\_db.facultie\_exists(textBox\_code\_facultie.Text))

{

MessageBox.Show("Такой код уже существует!", "специальность:ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

obj\_facultie\_db.add\_facultie(textBox\_code\_facultie.Text, textBox\_name\_facultie.Text, textBox\_short\_name.Text);

}

}

}

}

3.5 Тестирование программного обеспечения

Тестирование программного обеспечения важно для обеспечения его качества, надежности и безопасности. В результате тестирования можно обнаружить и исправить ошибки и недочеты в программе, предотвратить возможные сбои и проблемы при ее использовании.

Тестирование программного обеспечения - один из самых важных этапов в разработке любого программного продукта. Это процесс, направленный на выявление ошибок, дефектов и недочетов в работе программы, а также на проверку ее соответствия заявленным требованиям и функциональности.

В современном мире, где программное обеспечение играет огромную роль в повседневной жизни людей и в работе многих компаний, важно, чтобы оно было стабильным, надежным и безопасным. Именно поэтому роль тестирования программного обеспечения становится все более значимой.

Основные цели тестирования программного обеспечения — это выявление ошибок и дефектов, улучшение качества программного обеспечения. Недостаточно протестированное программное обеспечение может привести к серьезным последствиям, таким как сбои в работе, утечка конфиденциальной информации и потеря данных.

Для проведения тестирования программного обеспечения для учета абитуриентов разработан и реализован следующий план тестирования:

1. Подключение к БД

Если были введены и сохранены некорректные значения для подключения к БД или с нет соединения, то при запуске приложения выйдет MessageBox с уведомлением ошибки подключения, а также MessageBox с рекомендацией обратится к системному администратору техникума. На рисунках 13-14 показана ошибка подключения и рекомендация к дальнейшим действиям.

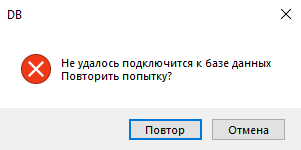


Рисунок 13 — неудачное подключение к БД

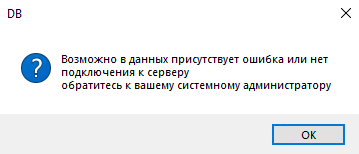


Рисунок 14 — рекомендация в случае неудачного подключения к БД

Если повторное подключение также оказалось неудачным, то приложение перенаправит пользователя на страницу настроек подключения к БД.

1. Добавление специальности

При попытке ввести пустые значения программа выдаст ошибку о заполнении данных. На рисунке 15 указана ошибка заполнения пустых данных при попытке создать новую специальность.

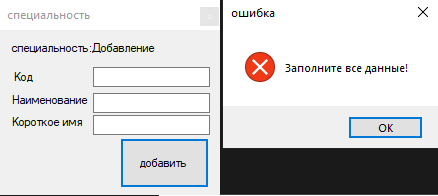


Рисунок 15 — ошибка добавления специальности

При попытке ввести значение специальности, которая уже присутствует в БД приведёт к ошибке. На рисунке 16 указана ошибка заполнения дубликата данных при попытке создать новую специальность.

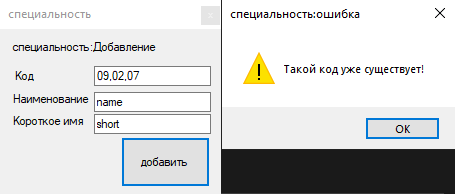


Рисунок 16 — ошибка дубликата кода специальности

Если при изменении данных специальности ничего не изменить, то это приведёт ошибке, что проиллюстрировано на рисунке 17. Ошибку можно вызвать, если значение уже имеется в БД, ошибка показана на рисунке 18.

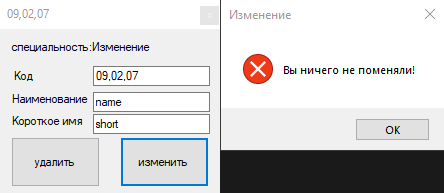


Рисунок 17 — Вызов ошибки из-за неизменённых значений

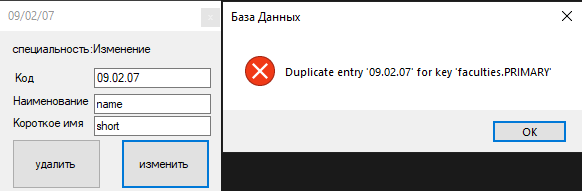


Рисунок 18 — ошибка дубликата кода

1. Редактирование балов абитуриента за экзамены

Если сумма баллов абитуриента превысит значение в 400, то программа запретит вносить изменения в БД. На рисунке 19 показана ошибка ввода баллов.

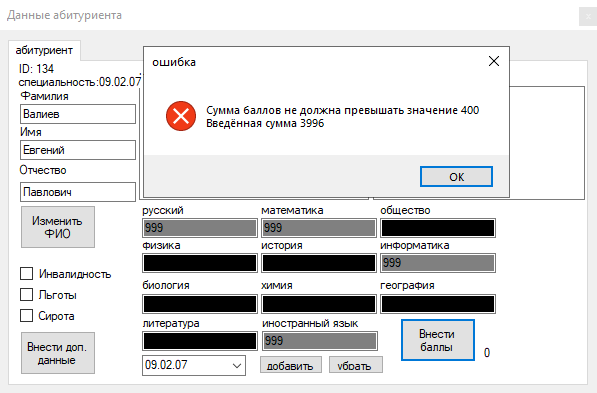


Рисунок 19 — ошибка ввода баллов

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано программное обеспечение для эффективного учета абитуриентов техникума. Программа обладает широким функционалом и позволяет автоматизировать рутинные процессы.

При разработке использовалась рабочая среда программного обеспечения Microsoft Visual Studio с использованием объектно-ориентированного языка программирования C# и системы управления базами данных MySQL. Программный продукт соответствует требованиям технического задания.

Программное обеспечение для учета абитуриентов представляет собой надежный инструмент для автоматизации бизнес-процессов и повышения эффективности работы техникума. Оно обладает высокой степенью функциональности, гибкости и масштабируемости, что делает его идеальным выбором для современных автосалонов.

В результате работы над проектом были достигнуты следующие цели:

1. Создание удобного и интуитивно понятного интерфейса для пользователей, что позволяет быстро и легко осуществлять учетность.

2. Реализация функционала для хранения и обработки абитуриентов и их личной информации.

3. Разработка возможности ведения БД.

В целом, разработка программного обеспечения для учета абитуриентов техникума была успешной и результативной.

Таким образом, в выпускной квалификационной работе в полном объеме решены все поставленные задачи.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://skillbox.ru/media/code/kak\_zashchitit\_bazu\_dannykh\_v\_c/ (19.05.24)
2. https://www.internet-technologies.ru/articles/obzor-konstruktora-jimdo.html#header-10435-1 (19.05.24)
3. https://metanit.com/sql/sqlserver/11.1.php (19.05.24)
4. https://ru.hexlet.io/qna/sql/questions/kak-vyzvat-protseduru-v-sql#:~:text=Для%20вызова%20процедуры%20в%20SQL,зависимости%20от%20используемой%20базы%20данных.&text=Также%20можно%20использовать%20операторы%20вызова,для%20работы%20с%20базами%20данных. (19.05.24)
5. https://sky.pro/wiki/sql/deklaratsiya-i-ispolzovanie-peremennykh-v-my-sql-primery/ (19.05.24)
6. https://sql-ex.ru/blogs/?/Rabota\_s\_hranimymi\_funkciJami\_v\_MySQL.html (19.05.24)
7. https://oracleplsql.ru/functions-mysql.html (19.05.24)
8. https://learn.microsoft.com/ru-ru/azure/mysql/single-server/connect-csharp (22.05.24)
9. https://metanit.com/sharp/tutorial/5.2.php (23.05.24)

ПРИЛОЖЕНИЕ

Приложение 1

**Алгоритм проектирование SQL запросов на создание базы данных**

-- MySQL dump 10.13 Distrib 8.0.36, for Win64 (x86\_64)

--

-- Host: localhost Database: trotsa

-- ------------------------------------------------------

-- Server version 8.0.36

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `applicant\_faculties`

--

DROP TABLE IF EXISTS `applicant\_faculties`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `applicant\_faculties` (

`id\_operation` int NOT NULL AUTO\_INCREMENT,

`ID\_applicant` int NOT NULL,

`code\_facultie` varchar(100) NOT NULL,

PRIMARY KEY (`id\_operation`),

KEY `facultie\_id` (`code\_facultie`),

KEY `applicant\_id` (`ID\_applicant`),

CONSTRAINT `applicant\_id` FOREIGN KEY (`ID\_applicant`) REFERENCES `applicants` (`ID\_applicant`),

CONSTRAINT `facultie\_id` FOREIGN KEY (`code\_facultie`) REFERENCES `faculties` (`code\_facultie`)

) ENGINE=InnoDB AUTO\_INCREMENT=26 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `applicant\_faculties`

--

LOCK TABLES `applicant\_faculties` WRITE;

/\*!40000 ALTER TABLE `applicant\_faculties` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `applicant\_faculties` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `applicants`

--

DROP TABLE IF EXISTS `applicants`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `applicants` (

`ID\_applicant` int NOT NULL AUTO\_INCREMENT,

`surname` varchar(50) NOT NULL,

`name` varchar(50) NOT NULL,

`patronymic` varchar(50) NOT NULL,

PRIMARY KEY (`ID\_applicant`)

) ENGINE=InnoDB AUTO\_INCREMENT=134 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `applicants`

--

LOCK TABLES `applicants` WRITE;

/\*!40000 ALTER TABLE `applicants` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `applicants` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `exam\_applicants`

--

DROP TABLE IF EXISTS `exam\_applicants`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `exam\_applicants` (

`id` int NOT NULL AUTO\_INCREMENT,

`ID\_applicant` int NOT NULL,

`russian` int DEFAULT '0',

`math` int DEFAULT '0',

`social\_science` int DEFAULT '0',

`physic` int DEFAULT '0',

`history` int DEFAULT '0',

`computer\_science` int DEFAULT '0',

`biology` int DEFAULT '0',

`chemistry` int DEFAULT '0',

`geography` int DEFAULT '0',

`literature` int DEFAULT '0',

`foreign\_language` int DEFAULT '0',

PRIMARY KEY (`id`),

KEY `code\_applicant\_idx` (`ID\_applicant`),

CONSTRAINT `code\_applicant` FOREIGN KEY (`ID\_applicant`) REFERENCES `applicants` (`ID\_applicant`)

) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `exam\_applicants`

--

LOCK TABLES `exam\_applicants` WRITE;

/\*!40000 ALTER TABLE `exam\_applicants` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `exam\_applicants` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `exam\_faculty`

--

DROP TABLE IF EXISTS `exam\_faculty`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `exam\_faculty` (

`id` int NOT NULL AUTO\_INCREMENT,

`code\_facultie` varchar(50) NOT NULL,

`russian` tinyint(1) DEFAULT '0',

`math` tinyint(1) DEFAULT '0',

`social\_science` tinyint(1) DEFAULT '0',

`physic` tinyint(1) DEFAULT '0',

`history` tinyint(1) DEFAULT '0',

`computer\_science` tinyint(1) DEFAULT '0',

`biology` tinyint(1) DEFAULT '0',

`chemistry` tinyint(1) DEFAULT '0',

`geography` tinyint(1) DEFAULT '0',

`literature` tinyint(1) DEFAULT '0',

`foreign\_language` tinyint(1) DEFAULT '0',

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `code\_faculty\_idx` (`code\_facultie`),

CONSTRAINT `code\_faculty` FOREIGN KEY (`code\_facultie`) REFERENCES `faculties` (`code\_facultie`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `exam\_faculty`

--

LOCK TABLES `exam\_faculty` WRITE;

/\*!40000 ALTER TABLE `exam\_faculty` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `exam\_faculty` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `faculties`

--

DROP TABLE IF EXISTS `faculties`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `faculties` (

`code\_facultie` varchar(50) NOT NULL,

`name\_facultie` varchar(50) NOT NULL,

`short\_name` varchar(50) NOT NULL,

PRIMARY KEY (`code\_facultie`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `faculties`

--

LOCK TABLES `faculties` WRITE;

/\*!40000 ALTER TABLE `faculties` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `faculties` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `personal\_data\_applicant`

--

DROP TABLE IF EXISTS `personal\_data\_applicant`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `personal\_data\_applicant` (

`ID\_pda` int NOT NULL AUTO\_INCREMENT,

`ID\_applicant` int NOT NULL,

`exam\_scores` int DEFAULT NULL,

`Disability` tinyint(1) DEFAULT '0',

`Privileges` tinyint(1) DEFAULT '0',

`Orphan` tinyint(1) DEFAULT '0',

`Documents\_Commentary` text,

`Commentary` tinytext,

PRIMARY KEY (`ID\_pda`),

KEY `ID\_applicant` (`ID\_applicant`),

CONSTRAINT `personal\_data\_applicant\_ibfk\_1` FOREIGN KEY (`ID\_applicant`) REFERENCES `applicants` (`ID\_applicant`)

) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `personal\_data\_applicant`

--

LOCK TABLES `personal\_data\_applicant` WRITE;

/\*!40000 ALTER TABLE `personal\_data\_applicant` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `personal\_data\_applicant` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `registrars`

--

DROP TABLE IF EXISTS `registrars`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `registrars` (

`ID\_registrar` int NOT NULL AUTO\_INCREMENT,

`Login\_registrar` varchar(15) NOT NULL,

`Password\_registrar` varchar(10) NOT NULL,

`Administrator` tinyint(1) DEFAULT NULL,

PRIMARY KEY (`ID\_registrar`),

UNIQUE KEY `Login\_registrar` (`Login\_registrar`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `registrars`

--

LOCK TABLES `registrars` WRITE;

/\*!40000 ALTER TABLE `registrars` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `registrars` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Dumping events for database 'trotsa'

--

--

-- Dumping routines for database 'trotsa'

--

/\*!50003 DROP FUNCTION IF EXISTS `authorization` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `authorization`(login\_reg varchar(15), pass\_reg varchar(10)) RETURNS tinyint(1)

DETERMINISTIC

BEGIN

DECLARE USER\_COUNTER INTEGER;

DECLARE OK BOOL;

select count(\*) INTO USER\_COUNTER from trotsa.registrars where Login\_registrar = login\_reg and Password\_registrar = pass\_reg;

IF (USER\_COUNTER = 0) then SET OK = FALSE;

ELSEIF (USER\_COUNTER = 1) THEN SET OK = TRUE;

END IF;

RETURN OK;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP FUNCTION IF EXISTS `exists\_applicants\_in\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `exists\_applicants\_in\_facultie`(code\_facultie\_param varchar(50), ID\_applicant\_param int) RETURNS tinyint(1)

DETERMINISTIC

BEGIN

declare count\_param int;

declare ok bool;

SELECT count(\*) into count\_param FROM trotsa.applicant\_faculties where ID\_applicant = ID\_applicant\_param and code\_facultie = code\_facultie\_param;

if(count\_param = 0) then set ok = false;

elseif(count\_param = 1) then set ok = true;

end if;

RETURN ok;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP FUNCTION IF EXISTS `facultie exist` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `facultie exist`(param\_code\_facultie varchar(50)) RETURNS int

DETERMINISTIC

BEGIN

DECLARE facultie\_COUNTER INTEGER;

DECLARE OK BOOL;

select count(\*) INTO facultie\_COUNTER from trotsa.faculties where code\_facultie = param\_code\_facultie;

IF (facultie\_COUNTER = 0) then SET OK = FALSE;

ELSEIF (facultie\_COUNTER = 1) THEN SET OK = TRUE;

END IF;

RETURN OK;

RETURN 1;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP FUNCTION IF EXISTS `facultie\_name` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `facultie\_name`(param\_code\_facultie varchar(50)) RETURNS varchar(50) CHARSET utf8mb4

DETERMINISTIC

BEGIN

declare nameFacultie varchar(50);

select name\_facultie into nameFacultie from trotsa.faculties where code\_facultie = param\_code\_facultie;

RETURN nameFacultie;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP FUNCTION IF EXISTS `facultie\_short\_name` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `facultie\_short\_name`(param\_code\_facultie varchar(50)) RETURNS varchar(50) CHARSET utf8mb4

DETERMINISTIC

BEGIN

declare nameFacultie varchar(50);

select short\_name into nameFacultie from trotsa.faculties where code\_facultie = param\_code\_facultie;

RETURN nameFacultie;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP FUNCTION IF EXISTS `get\_current\_facultie\_applicant` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `get\_current\_facultie\_applicant`(ID\_applicant\_param int) RETURNS varchar(50) CHARSET utf8mb4

DETERMINISTIC

BEGIN

declare code\_ret varchar(50);

SELECT code\_facultie into code\_ret FROM trotsa.applicant\_faculties where ID\_applicant = ID\_applicant\_param;

RETURN code\_ret;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP FUNCTION IF EXISTS `IsUserAdmin` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `IsUserAdmin`(Login\_registrar\_param varchar(50)) RETURNS tinyint(1)

DETERMINISTIC

BEGIN

declare admin\_param tinyint(1);

SELECT Administrator into admin\_param FROM trotsa.registrars where Login\_registrar = Login\_registrar\_param;

RETURN admin\_param;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP FUNCTION IF EXISTS `search\_login\_registar` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` FUNCTION `search\_login\_registar`(login\_reg varchar(15)) RETURNS tinyint(1)

DETERMINISTIC

BEGIN

DECLARE USER\_COUNTER INTEGER;

DECLARE OK BOOL;

select count(\*) INTO USER\_COUNTER from trotsa.registrars where Login\_registrar = login\_reg;

IF (USER\_COUNTER = 0) then SET OK = FALSE;

ELSEIF (USER\_COUNTER = 1) THEN SET OK = TRUE;

END IF;

RETURN OK;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `access\_to\_exams` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `access\_to\_exams`(code\_facultie\_param varchar(50))

BEGIN

SELECT \* FROM trotsa.exam\_faculty where code\_facultie =code\_facultie\_param;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `add\_applicant` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `add\_applicant`(arg\_name varchar(10), arg\_surname varchar(10), arg\_patronymic varchar(10))

BEGIN

declare last\_id int;

INSERT INTO `trotsa`.`applicants` (`name`, `surname`, `patronymic`) VALUES (arg\_name, arg\_surname, arg\_patronymic);

set last\_id = last\_insert\_id();

INSERT INTO `trotsa`.`personal\_data\_applicant` (`ID\_applicant`, `exam\_scores`) VALUES (last\_id, 0);

INSERT INTO `trotsa`.`exam\_applicants` (`ID\_applicant`) VALUES (last\_id);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `add\_applicant\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `add\_applicant\_facultie`(ID\_applicant\_param int, code\_facultie\_param varchar(50))

BEGIN

DELETE FROM `trotsa`.`applicant\_faculties` WHERE (`code\_facultie` != code\_facultie\_param);

INSERT INTO `trotsa`.`applicant\_faculties` (`ID\_applicant`, `code\_facultie`) VALUES (ID\_applicant\_param, code\_facultie\_param);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `add\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `add\_facultie`(code\_facultie varchar(50), name\_facultie varchar(50), short\_name\_facultie varchar(50))

BEGIN

INSERT INTO `trotsa`.`faculties` (`code\_facultie`, `name\_facultie`, `short\_name`) VALUES (code\_facultie, name\_facultie, short\_name\_facultie);

INSERT INTO `trotsa`.`exam\_faculty` (`code\_facultie`, `russian`, `math`, `social\_science`, `physic`, `history`, `computer\_science`, `biology`, `chemistry`, `geography`, `literature`, `foreign\_language`) VALUES (code\_facultie, '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0');

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `change\_code` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `change\_code`(old\_code varchar(50), new\_code varchar(50))

BEGIN

declare name\_facultie\_param varchar(50);

declare short\_name\_facultie\_param varchar(50);

SELECT name\_facultie into name\_facultie\_param FROM trotsa.faculties where code\_facultie = old\_code;

SELECT short\_name into short\_name\_facultie\_param FROM trotsa.faculties where code\_facultie = old\_code;

INSERT INTO `trotsa`.`faculties` (`code\_facultie`, `name\_facultie`, `short\_name`) VALUES (new\_code, name\_facultie\_param, short\_name\_facultie\_param);

UPDATE `trotsa`.`applicant\_faculties` SET `code\_facultie` = new\_code WHERE (`code\_facultie` = old\_code);

UPDATE `trotsa`.`exam\_faculty` SET `code\_facultie` = new\_code WHERE (`code\_facultie` = old\_code);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `change\_requirements\_exam\_faculty` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `change\_requirements\_exam\_faculty`(code\_facultie\_param varchar(50),

`russian\_param` tinyint(1), `math\_param` tinyint(1), `social\_science\_param` tinyint(1),

`physic\_param` tinyint(1), `history\_param` tinyint(1), `computer\_science\_param` tinyint(1),

`biology\_param` tinyint(1), `chemistry\_param` tinyint(1), `geography\_param` tinyint(1),

`literature\_param` tinyint(1), `foreign\_language\_param` tinyint(1))

BEGIN

UPDATE `trotsa`.`exam\_faculty` SET `russian` = russian\_param, `math` = math\_param, `social\_science` = social\_science\_param,

`physic` = physic\_param, `history` = history\_param, `computer\_science` = computer\_science\_param,

`biology` = biology\_param, `chemistry` = chemistry\_param, `geography` = geography\_param,

`literature` = literature\_param, `foreign\_language` = foreign\_language\_param WHERE (code\_facultie = code\_facultie\_param);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `delete\_applicant` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_applicant`(ID\_applicant\_param int)

BEGIN

DELETE FROM `trotsa`.`applicant\_faculties` WHERE (`ID\_applicant` = ID\_applicant\_param);

DELETE FROM `trotsa`.`personal\_data\_applicant` WHERE (`ID\_applicant` = ID\_applicant\_param);

DELETE FROM `trotsa`.`exam\_applicants` WHERE (`ID\_applicant` = ID\_applicant\_param);

DELETE FROM `trotsa`.`applicants` WHERE (`ID\_applicant` = ID\_applicant\_param);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `delete\_applicant\_from\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_applicant\_from\_facultie`(ID\_applicant\_param int, code\_facultie\_param varchar(50))

BEGIN

DELETE FROM `trotsa`.`applicant\_faculties` WHERE (`ID\_applicant` = ID\_applicant\_param and code\_facultie = code\_facultie\_param);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `delete\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_facultie`(delete\_code varchar(50))

BEGIN

DELETE FROM `trotsa`.`applicant\_faculties` WHERE (code\_facultie = delete\_code);

DELETE FROM `trotsa`.`faculties` WHERE (`code\_facultie` = delete\_code);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `delete\_old\_code\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_old\_code\_facultie`(code\_facultie\_param varchar(50))

BEGIN

SET FOREIGN\_KEY\_CHECKS=0;

DELETE FROM `trotsa`.`faculties` WHERE (`code\_facultie` = code\_facultie\_param) limit 1;

SET FOREIGN\_KEY\_CHECKS=1;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `edit\_fio\_applicants` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `edit\_fio\_applicants`(id int, surname\_param varchar(50), `name\_param` varchar(50), patronymic\_param varchar(50))

BEGIN

UPDATE `trotsa`.`applicants` SET `surname` = surname\_param, `name` = `name\_param`, `patronymic` = patronymic\_param WHERE (`ID\_applicant` = id);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `faculties\_of\_applicant` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `faculties\_of\_applicant`(code\_param int)

BEGIN

select code\_facultie, name\_facultie, short\_name FROM trotsa.applicant\_faculties

join faculties on faculties.code\_facultie = `faculties\_of\_applicant`.code\_facultie

where ( code\_param = code\_facultie);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `get\_applicants` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_applicants`(counter\_from int, counter\_to int)

BEGIN

SELECT applicants.ID\_applicant as `Идентификатор`, applicants.surname as `Фамилия`, applicants.`name` as `Имя`, applicants.patronymic as `Отчество`

FROM trotsa.applicants

join trotsa.personal\_data\_applicant on trotsa.applicants.ID\_applicant = trotsa.personal\_data\_applicant.ID\_applicant

order by exam\_scores desc

limit counter\_from, counter\_to;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `get\_applicants\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_applicants\_facultie`(facultie varchar(50), counter\_from int, counter\_to int)

BEGIN

SELECT applicants.ID\_applicant as `Идентификатор`, applicants.surname as `Фамилия`, applicants.`name` as `Имя`, applicants.patronymic as `Отчество`

FROM trotsa.applicants

join trotsa.applicant\_faculties on trotsa.applicants.ID\_applicant = trotsa.applicant\_faculties.ID\_applicant and trotsa.applicant\_faculties.code\_facultie = facultie

join trotsa.personal\_data\_applicant on trotsa.applicants.ID\_applicant = trotsa.personal\_data\_applicant.ID\_applicant

#group by applicants.ID\_applicant

order by exam\_scores desc

limit counter\_from, counter\_to;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `get\_exam\_requirements` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_exam\_requirements`(code\_facultie\_param varchar(50))

BEGIN

SELECT russian, math, social\_science, physic, `history`, computer\_science, biology, chemistry, geography, literature, foreign\_language FROM trotsa.exam\_faculty

where code\_facultie = code\_facultie\_param;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `update\_code\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_code\_facultie`(old\_code\_facultie varchar(50), new\_code\_facultie varchar(50))

BEGIN

INSERT INTO `faculties` (`code\_facultie`, name\_facultie, short\_name)

VALUES (new\_code\_facultie, '','');

UPDATE `applicant\_faculties` SET `code\_facultie` = new\_code\_facultie WHERE (`code\_facultie` = old\_code\_facultie);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 DROP PROCEDURE IF EXISTS `update\_data\_facultie` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_data\_facultie`(code\_facultie varchar(50), name\_facultie varchar(50), short\_facultie varchar(50))

BEGIN

UPDATE `trotsa`.`faculties` SET `name\_facultie` = name\_facultie, `short\_name` = short\_facultie WHERE (`code\_facultie` = code\_facultie);

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2024-06-12 1:06:01