

**Колледж космического машиностроения и технологий**

**ОТЧЕТ ПО ЭКЗАМЕНУ**

по профессиональному модулю ПМ.01 «Эксплуатация и модификация информационных систем» специальности 09.02.04 «Информационные системы (по отраслям)»

Экзаменационный билет №16

Обучающегося 3 курса группы ИС2-20 формы обучения очной

|  |
| --- |
| **Пономарев Кирилл Павлович** |
|  |

Дата прохождения экзамена «30» июня 2023 г.

**Оглавление**

[1. Экзаменационный билет 3](#_Toc139072668)

[2. Жизненный цикл ИС 4](#_Toc139072669)

[3. Функциональные и технические требования ИС 7](#_Toc139072670)

[4. Сценарий работы системы 9](#_Toc139072671)

[5. Структура базы данных 14](#_Toc139072672)

[6. Внешний вид 17](#_Toc139072673)

[7. Инструментальные средства разработки ИС 20](#_Toc139072674)

[8. Результаты разработки 21](#_Toc139072675)

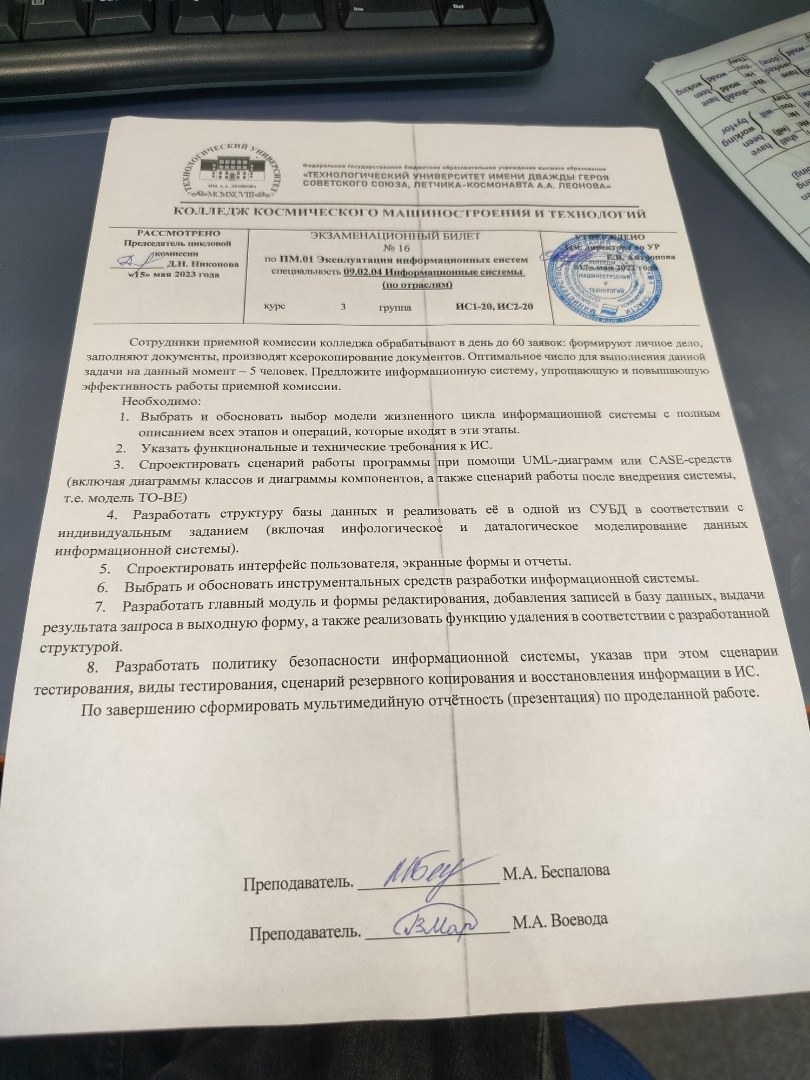
[8.1 Выполнение требований 21](#_Toc139072676)

[9. Политика безопасности ИС 22](#_Toc139072677)

[9.1 Резервное копирование 22](#_Toc139072678)

[10. Результаты выполнения экзаменационного билета 23](#_Toc139072679)

1. Экзаменационный билет



1. Жизненный цикл ИС

Под моделью жизненного цикла (ЖЦ) понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики ИС и специфики условий, в которых последняя создается и функционирует.

Существуют три основные модели ЖЦ информационных систем:

* Каскадная (рисунок 1) – её основной характеристикой является разбиение всей разработки на этапы, при этом переход на следующий этап происходит только после полного завершения работ на текущем.

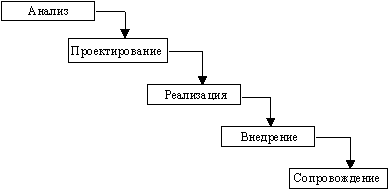


Рисунок 1. Каскадная модель ЖЦ

* Поэтапная (рисунок 2) – модель с промежуточным контролем. В поэтапной модели с промежуточным контролем разработка ПО ведётся итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют уменьшить трудоёмкость процесса разработки по сравнению с каскадной моделью. Время жизни каждого из этапов растягивается на весь период разработки.

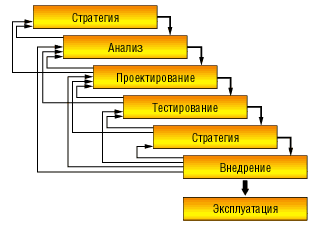


Рисунок 2. Каскадная модель ЖЦ

* Спиральная (рисунок 3) – модель ЖЦ, в которой особое внимание уделяется начальным этапам разработки – выработке стратегии, анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования). Каждый виток спирали предполагает создание фрагмента (компонента) или версии программного продукта. На них уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

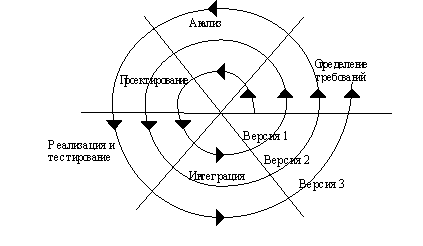


Рисунок 3. Спиральная модель ЖЦ

В ходе прохождения экзамена, была выбрана поэтапная модель ЖЦ. Данный выбор обоснован, наличием межэтапных корректировок в процессе разработки, что позволяет уменьшить её время и трудоемкость.

Во время выбора ЖЦ для разрабатываемой ИС, были определены, следующие этапы:

* Анализ – детальное изучение выданного задания на разработку;
* Проектирование – создание системы:
  + Основа – создание скелета, будущей системы на основе разработанных моделей базы данных (рисунок 11 и 12);
  + Программирование – написание автоматизированной части системы, что позволит повысить эффективность её работы и уменьшить затрачиваемую трудоёмкость при использовании;
  + Внешний вид – детальная проработка интерфейса системы, её форм, а также создание схемы компоновки данных для удобного предоставления информации по оплате аренды в отчёте;
* Тестирование – проведение мероприятий по тестированию основных модулей системы на наличие ошибок с применением тест-кейсов для наибольшей эффективности;

1. Функциональные и технические требования ИС

Согласно, выданному заданию были определены следующие функциональные требования к разрабатываемой ИС:

1. Регистрация абитуриентов:
   * Система должна предоставлять возможность абитуриентам регистрироваться в ИС приемной комиссии путем заполнения необходимых данных и создания учетной записи.
2. Подача заявлений:
   * Система должна позволять абитуриентам подавать заявления на поступление в электронном формате.
   * Система должна предоставлять возможность прикрепления необходимых документов к заявлению.
3. Обработка заявлений:
   * Система должна обеспечивать автоматическую обработку и анализ заявлений и документов, включая проверку соответствия требованиям и формирование отчетов о результатах обработки.
   * Система должна поддерживать процесс рассмотрения заявлений и принятия решений о приеме абитуриентов.
4. Уведомления и коммуникация:
   * Система должна автоматически уведомлять абитуриентов о результатах рассмотрения и принятых решениях через различные каналы связи, такие как электронная почта или SMS.
   * Система должна предоставлять возможность для обратной связи и коммуникации с абитуриентами, например, через систему обращений или онлайн-консультации.

Технические требования:

1. Платформа и архитектура:
   * Система должна быть разработана на определенной платформе, такой как веб-приложение или клиент-серверная архитектура.
   * Система должна быть совместима с используемыми операционными системами и базами данных.
2. Безопасность:
   * Система должна обеспечивать защиту данных абитуриентов и конфиденциальность информации.
   * Система должна иметь механизмы аутентификации и авторизации для контроля доступа пользователей.
3. Масштабируемость и производительность:
   * Система должна быть способна обрабатывать большое количество заявлений и данных, обеспечивая высокую производительность и отзывчивость.
   * Система должна быть масштабируемой, чтобы удовлетворить потребности растущего числа абитуриентов и объема данных.
4. Резервное копирование и восстановление:
   * Система должна поддерживать резервное копирование данных и возможность восстановления в случае сбоев или потери данных.
   * Система должна иметь процедуры и механизмы для регулярного резервного копирования и проверки целостности данных.
5. Сценарий работы системы

Общее описание:

На основе этого была составлена UML-диаграмма последовательности (рисунок 7), как пример, отражающий логику работы одной из системы.

UML, или Unified Modeling Language, — унифицированный язык моделирования. Это графический язык, который с помощью диаграмм и схем описывает разнообразные процессы и структуры.

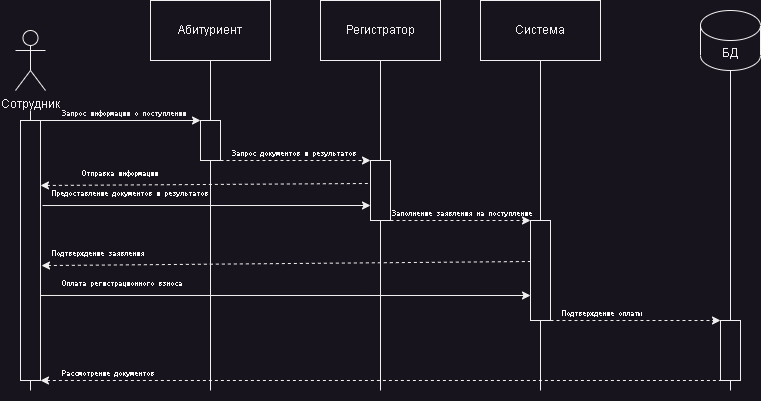


Рисунок 7. UML-диаграмма последовательности. Приемная Комиссия

Регистратор (Registrar) отправляет запрошенную информацию абитуриенту.

Абитуриент (Applicant) запрашивает у Регистратора (Registrar) список документов и требования для поступления, а также информацию о результате рассмотрения заявления.

Регистратор (Registrar) предоставляет абитуриенту необходимые документы и информацию о результате рассмотрения заявления.

Абитуриент (Applicant) заполняет заявление на поступление.

Регистратор (Registrar) подтверждает заявление абитуриента.

Абитуриент (Applicant) осуществляет оплату регистрационного взноса.

Регистратор (Registrar) подтверждает оплату регистрационного взноса.

Приемная комиссия рассматривает предоставленные абитуриентом документы.

Приемная комиссия отправляет абитуриенту уведомление о результате рассмотрения его заявления.

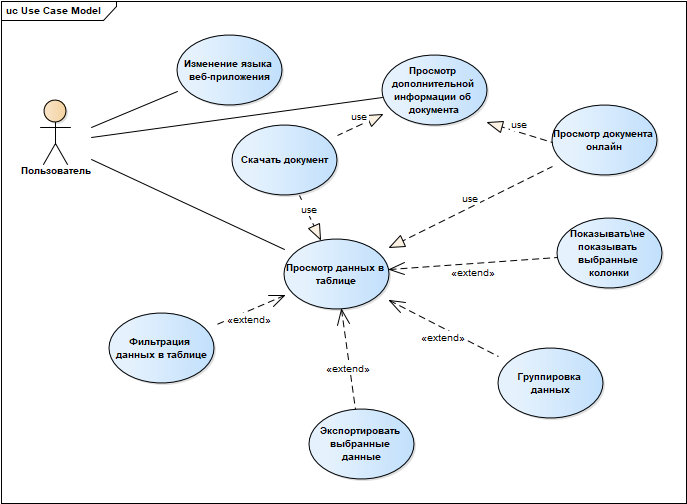


Рисунок 8. Модель TO-BE

Для отображения логики между структурой данных, описанной в пункте 5 «Структура базы данных» и сценария работы программы (по большей части её автоматизации), была составлена диаграмма классов (рисунок 9).

На данной диаграмме отображены классы сущностей, используемых в системе, а также их процедуры, относящиеся к блоку автоматизации на диаграмме компонентов (рисунок 10).

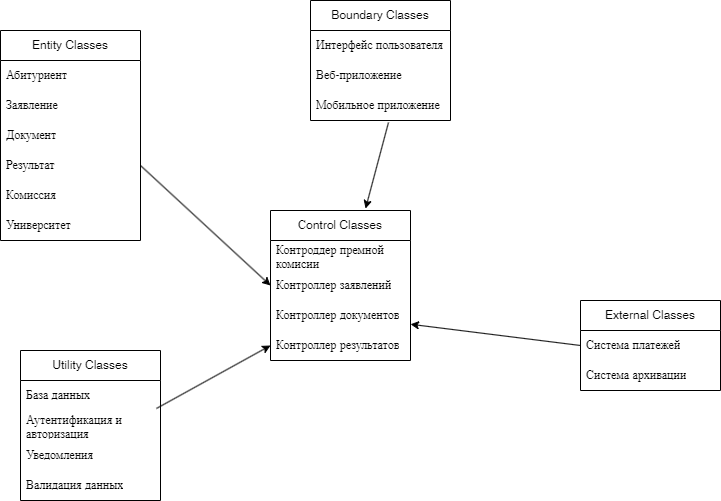


Рисунок 9. Диаграмма классов

Для наглядности общего описания в начале данного пункта была составлена диаграмма компонентов (рисунок 10).

В данном примере используются различные категории классов для обозначения их роли и функциональности в системе. Entity Classes представляют основные сущности, такие как абитуриенты, заявления, документы и т.д. Boundary Classes отвечают за взаимодействие с внешними системами или пользователем, например, интерфейс пользователя или веб-приложение. Control Classes управляют бизнес-логикой системы, такими как контроллер приемной комиссии, контроллер заявлений и т.д. Utility Classes представляют вспомогательные функции, такие как работа с базой данных, аутентификация, уведомления и валидация данных. External Classes представляют внешние системы, с которыми система приемной комиссии взаимодействует, например, система платежей или система архивации.

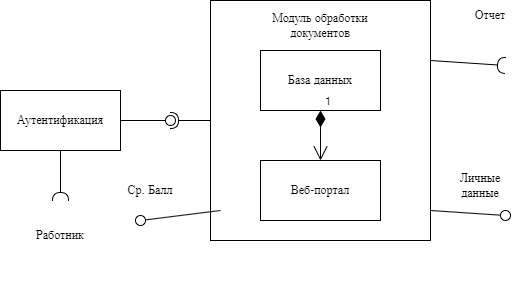


Рисунок 10. Диаграмма компонентов

Веб-портал: Компонент, отвечающий за интерфейс взаимодействия с абитуриентами. Здесь абитуриенты могут заполнять заявления, загружать документы, получать информацию о поступлении и общаться с приемной комиссией.

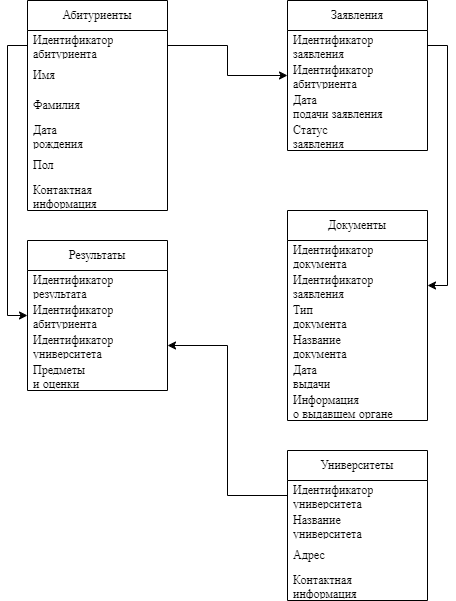
База данных: Компонент, который хранит информацию о заявлениях абитуриентов, документах, результатах рассмотрения и других сущностях, связанных с приемной комиссией.

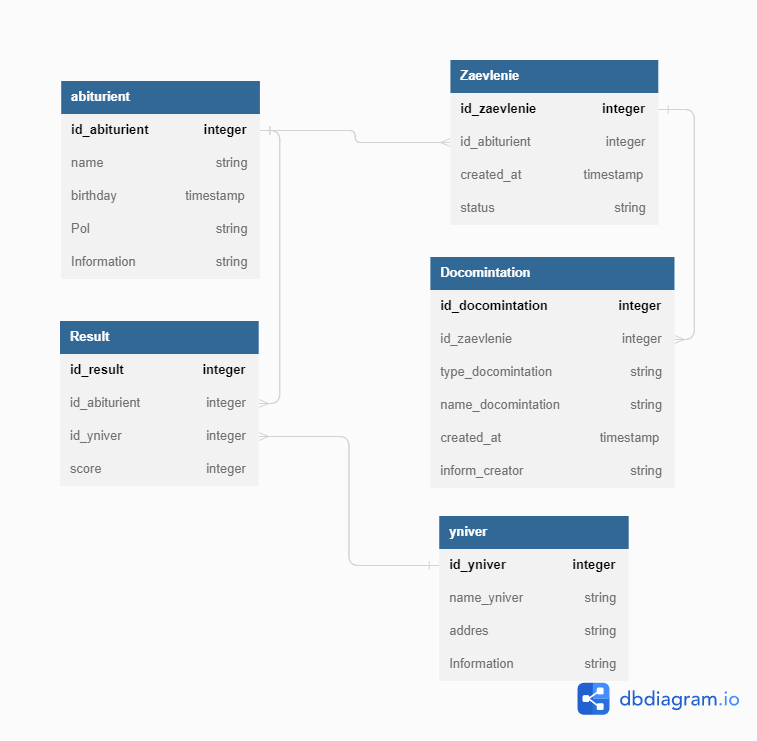
Аутентификация: Компонент, отвечающий за проверку подлинности абитуриентов при доступе к веб-порталу. Здесь осуществляется аутентификация абитуриентов и управление их учетными записями.

Модуль обработки документов: Компонент, который выполняет обработку и анализ документов, предоставленных абитуриентами. Он может включать функциональности распознавания текста, проверки соответствия требованиям, а также генерации отчетов для приемной комиссии.

Это примерная UML-диаграмма компонентов для приемной комиссии. Каждый компонент выполняет определенные функции и взаимодействует с другими компонентами для обеспечения работы процесса приема абитуриентов. Обратите внимание, что эта диаграмма является обобщенной моделью, и реальная система приемной комиссии может иметь дополнительные компоненты в зависимости от ее конкретных потребностей и функциональности.

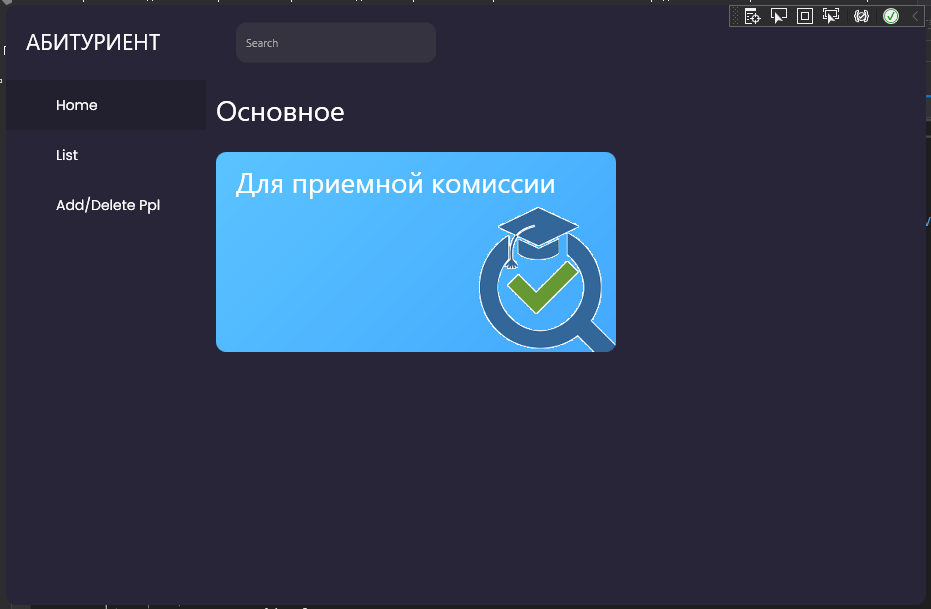
1. Структура базы данных

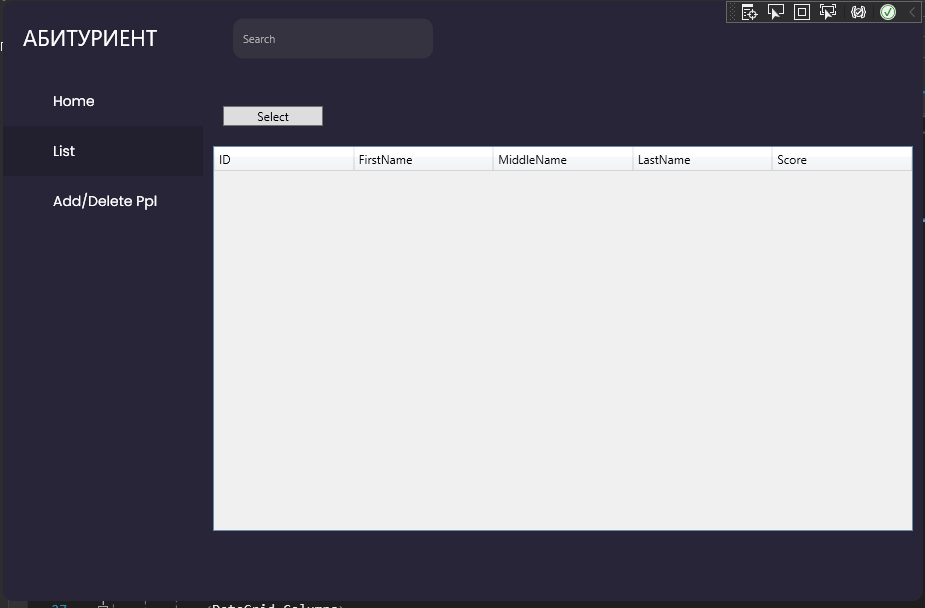


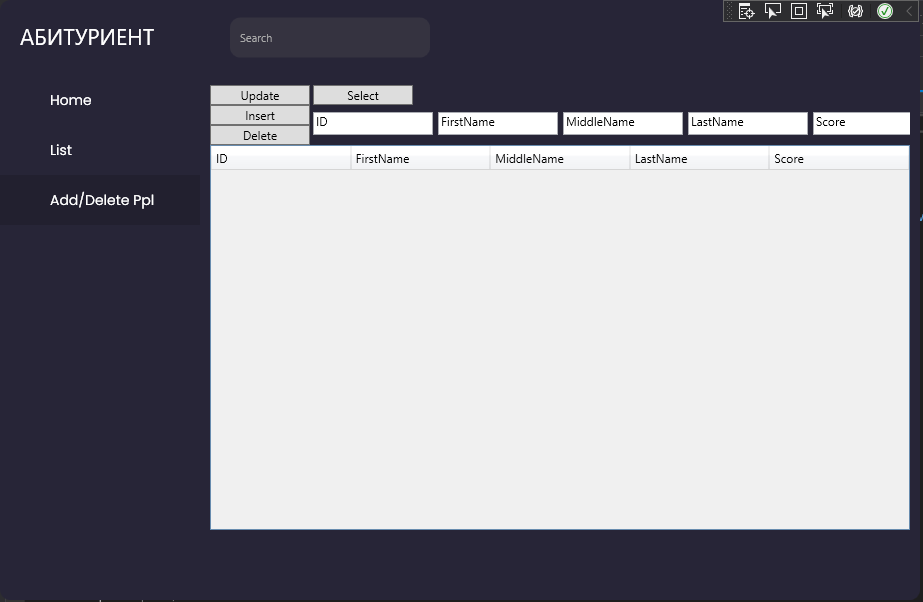


Пример структуры базы данных:

1. Таблица "Абитуриенты":
   * Идентификатор абитуриента (Primary Key)
   * Имя
   * Фамилия
   * Дата рождения
   * Пол
   * Контактная информация (адрес, телефон, электронная почта)
2. Таблица "Заявления":
   * Идентификатор заявления (Primary Key)
   * Идентификатор абитуриента (Foreign Key)
   * Дата подачи заявления
   * Статус заявления (например, "рассматривается", "принято", "отклонено")
3. Таблица "Документы":
   * Идентификатор документа (Primary Key)
   * Идентификатор заявления (Foreign Key)
   * Тип документа (например, аттестат, паспорт, сертификат)
   * Название документа
   * Дата выдачи
   * Информация о выдавшем органе
4. Таблица "Результаты":
   * Идентификатор результата (Primary Key)
   * Идентификатор абитуриента (Foreign Key)
   * Идентификатор университета (Foreign Key)
   * Предметы и оценки (например, математика - 90, физика - 85)
5. Таблица "Университеты":
   * Идентификатор университета (Primary Key)
   * Название университета
   * Адрес
   * Контактная информация (телефон, электронная почта)
6. Внешний вид







Принципы для создания приятного и функционального внешнего вида программы для приемной комиссии:

1. Простой и интуитивно понятный интерфейс: Обеспечьте простоту и легкость использования программы. Интерфейс должен быть понятным для пользователей, даже для тех, кто не имеет опыта работы с подобными системами.
2. Структурированный макет: Организуйте элементы интерфейса таким образом, чтобы пользователи могли легко найти нужную информацию и выполнять необходимые действия. Разделите информацию на логические блоки и используйте понятные заголовки и разделители.
3. Четкая навигация: Обеспечьте удобную навигацию по программе, чтобы пользователи могли быстро перемещаться между различными разделами и функциями. Используйте наглядные элементы навигации, такие как меню, вкладки или панели инструментов.
4. Современный дизайн и согласованность: Учтите современные тренды дизайна, чтобы создать актуальный и эстетически приятный внешний вид программы. Обратите внимание на использование подходящих цветовых схем, шрифтов, иконок и элементов стиля. Также важно сохранять согласованность дизайна на всех страницах и экранах программы.
5. Респонсивный дизайн: если ваша программа будет использоваться на различных устройствах (например, настольных компьютерах, планшетах, смартфонах), рекомендуется разработать респонсивный дизайн. Это позволит программе адаптироваться к разным размерам экранов и устройствам, обеспечивая удобство использования для пользователей.
6. Хорошо оформленные формы и элементы ввода: если в программе присутствуют формы для заполнения данных, уделите внимание их оформлению. Используйте понятные метки для полей ввода, подсказки и проверки ввода данных, чтобы помочь пользователям заполнить формы корректно.
7. Информационные сообщения и уведомления: включите информационные сообщения и уведомления, чтобы предоставлять пользователю обратную связь о выполняемых действиях, статусе процессов или сообщениях об ошибках. Уведомления должны быть понятными и наглядными, помогая пользователям ориентироваться в системе.
8. Персонализация и настройки: рассмотрите возможность предоставления настройки пользовательского интерфейса, таких как выбор языка, темы оформления или настроек уведомлений. Это позволит пользователям настроить программу под свои предпочтения.
9. Инструментальные средства разработки ИС

Система разрабатывалась на платформе С# по причине наличия существенных плюсов у данной платформы:

* Гибкая настройка UI;
* Консоль для отладки;
* Настройка отображения списков «под себя» и сохранение этих настроек;

Так же существенным плюсом является простота подключения С# к самостоятельной базе данных (PostgreSQL, MySQL и др.) с использованием сторонних СУБД. При необходимости это делается легко и быстро с помощью официальных плагинов.

1. Результаты разработки

В результате выполнения экзаменационного задания была разработана полноценная информационная система для приемной комиссии.

8.1 Выполнение требований

Реализованы все функциональные требования к системе:

* Система должна позволять абитуриентам зарегистрироваться и создать учетные записи для доступа к функциональности системы.
* Система должна обеспечивать аутентификацию абитуриентов для защиты их личной информации и предотвращения несанкционированного доступа.
* Система должна предоставлять возможность абитуриентам заполнять и подавать заявления на поступление в электронном формате.
* Система должна позволять абитуриентам загружать и прикреплять необходимые документы к заявлению, такие как свидетельства о результатах обучения, рекомендательные письма и т.д.
* Система должна автоматически обрабатывать и анализировать заявления и предоставленные документы, включая проверку соответствия требованиям и формирование отчетов о результатах обработки.
* Система должна поддерживать процесс рассмотрения заявлений и принятия решений о приеме абитуриентов.

1. Политика безопасности ИС

9.1 Резервное копирование

Для резервного копирования базы данных PostgreSQL можно использовать несколько подходов. Некоторые из них:

pg\_dump: pg\_dump — это утилита командной строки, входящая в состав PostgreSQL, которая позволяет создавать резервные копии базы данных. Вы можете использовать следующую команду для создания резервной копии базы данных:

pg\_dumpall: pg\_dumpall — это утилита командной строки, которая позволяет создавать резервные копии всех баз данных в PostgreSQL, включая системные таблицы.

Continuous Archiving and Point-in-Time Recovery (PITR): Этот метод позволяет создавать периодические резервные копии и восстанавливать базу данных до определенного момента времени. Он включает настройку архивации журналов транзакций и использование инструментов восстановления, таких как pg\_basebackup и pg\_wal.

Этот подход обеспечивает более гранулированный уровень восстановления и защиту от потери данных в случае сбоев. Однако он требует более сложной настройки и администрирования.

Важно отметить, что резервное копирование должно выполняться регулярно и сохраняться в надежном месте, отделенном от основной системы баз данных. Также рекомендуется тестировать процесс восстановления из резервной копии, чтобы убедиться в его работоспособности в случае необходимости восстановления данных.

1. Результаты выполнения экзаменационного билета

Пункты экзаменационного задания выполнены:

1. Выбор модели ЖЦ и обоснование выбора описаны в пункте 2 «Жизненный цикл ИС»;
2. Требования к ИС описаны в пункте 3 «Функциональные и технические требования ИС»;
3. UML-диаграммы, модель TO-BE и их описание описано в пункте 4 «Сценарий работы системы»;
4. Структура базы данных (включая построение инфологической и даталогической моделей) описаны в пункте 5 «Структура базы данных»;
5. Интерфейса пользователя, экранные формы и отчёт описаны в пункте 6 «Внешний вид»;
6. Выбор и обоснование выбора средств проектирование описаны в пункте 7 «Инструментальные средства разработки ИС»;
7. Все описанные требования 7 пункта к разработке описаны в пункте документа 8 «Результаты разработки»;
8. Политика безопасности, включающая тестирование, резервирование и восстановление описана в пункте 9 «Политика безопасности ИС».
9. https://github.com/tom228studio/Qualification\_exam\_PM\_01