Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчёт**

по предмету «Проектирование программного обеспечения»

Лабораторная работа №5

«Проектирование информационных систем»

Студент: Лопатнюк П.В.

ФИТ 3 курс 1 группа

Преподаватель: Якубенко К. Д

Минск 2024

# Теоретические сведения

К структурным диаграммам UML 2.0 относятся такие виды, как:

* Диаграмма классов: моделирует статическую структуру системы, описывая классы, их атрибуты, методы и связи (ассоциации, зависимости, наследование).
* Диаграмма объектов: фиксирует состояние объектов системы на определенный момент времени, иллюстрируя взаимодействие между экземплярами.
* Диаграмма компонентов: описывает модульную архитектуру системы, связи и интерфейсы между компонентами.
* Диаграмма развертывания: демонстрирует, как программное обеспечение развёрнуто на физической инфраструктуре, включая узлы и артефакты.
* Диаграмма пакетов: упрощает понимание системы через группировку элементов в пакеты.
* Диаграмма композиционной структуры: показывает внутреннюю организацию класса или компонента, включая взаимодействие его частей.
* Диаграмма профилей: адаптирует UML для использования в конкретных областях через добавление пользовательских стереотипов и тегированных значений.

Каждая из структурных диаграмм имеет своё предназначение. Например, диаграмма классов используется для моделирования статической структуры системы, включая классы, их атрибуты, методы и взаимосвязи. Диаграмма объектов фиксирует конкретное состояние объектов системы в определённый момент времени, что полезно для анализа и демонстрации динамических аспектов. Диаграмма компонентов иллюстрирует модульную архитектуру системы, показывая взаимодействие между её основными частями. Диаграмма развертывания позволяет описывать, как программные компоненты размещаются на физической инфраструктуре, а диаграмма пакетов упрощает понимание системы через группировку связанных элементов. Диаграмма композиционной структуры помогает детализировать внутреннее устройство компонентов или классов, показывая их составные элементы и связи. Наконец, диаграмма профилей адаптирует UML для использования в конкретных областях.

Диаграммы классов строятся с использованием таких элементов, как прямоугольники, представляющие классы. Прямоугольники делятся на три секции: имя класса указывается в верхней части, атрибуты размещаются в средней секции, а методы описываются в нижней. Связи между классами:

−Ассоциация — линия между классами, которая обозначает, что один класс взаимодействует с другим. Может быть направленной (стрелка) или недиректной.

−Агрегация — полая ромбовидная стрелка, указывающая на «целое» и «части» (например, класс «Автомобиль» состоит из класса «Колесо»).

−Композиция — закрашенная ромбовидная стрелка, показывающая, что «часть» не может существовать без «целого» (например, «Комната» является частью «Дома»).

−Наследование (генерализация) — линия с треугольной стрелкой, указывающая на родительский класс, от которого наследуются дочерние классы.

−Зависимость — пунктирная линия со стрелкой, обозначающая, что один класс зависит от другого (например, использование одного класса в качестве параметра метода другого класса).

Интерфейс — изображается как прямоугольник с надписью <<interface>> или как круг (класс реализует интерфейс). Абстрактные классы и методы — обозначаются курсивом.

Расширения диаграмм UML позволяют адаптировать стандартные модели под специфические потребности различных отраслей и задач. Это помогает делать диаграммы более выразительными и удобными для понимания в узкоспециализированных областях, таких как медицина или финансовый сектор. С помощью расширений можно:

* Добавлять новые семантические элементы через профили, стереотипы и тегированные значения.
* Моделировать системы в специфических отраслях, таких как медицина, автомобильная промышленность или финансовый сектор.
* Упрощать использование UML для сложных систем, делая диаграммы более понятными для узкоспециализированных команд.

Стереотип в UML представляет собой механизм расширения, который позволяет уточнять смысл стандартных элементов. Например, можно добавить стереотип <<Entity>>, чтобы обозначить классы, представляющие сущности базы данных. В диаграммах стереотипы заключаются в угловые скобки и помогают выделить важные аспекты модели. Тегированное значение — это способ дополнить элементы UML дополнительными свойствами в формате "ключ-значение". Например, для класса можно указать такие значения, как автор или версия {Author=John Doe, Version=1.2}. Эти механизмы делают модели более гибкими и применимыми к конкретным задачам.

# Постановка задачи

Задачей данной работы является создание подробной диаграммы классов для программы «Интерфейс электронной почты», которая отразит структуру системы, взаимосвязи между её основными элементами, а также ключевые атрибуты и методы, необходимые для реализации функционала. Диаграмма должна учитывать роли пользователей (гость, пользователь, системный администратор), специфику их действий в системе и взаимосвязь между различными компонентами приложения.

Диаграмма классов — это один из центральных инструментов проектирования в UML, который обеспечивает наглядное представление статической архитектуры системы. Она используется для описания основных сущностей, их свойств и поведения, а также для отображения связей между ними. В данном проекте диаграмма классов выполняет следующие функции:

* Отражение структуры приложения: Диаграмма показывает, как организованы классы, включая их атрибуты, методы и наследуемые свойства.
* Визуализация взаимодействий: Упрощает понимание связей между классами, таких как ассоциации, наследование и композиция.
* Поддержка разработки: Помогает проектировщикам и разработчикам анализировать и реализовывать логику приложения.
* Формализация модели: Обеспечивает основу для создания связного, структурированного кода и подготовки к последующим доработкам системы.

Основной целью является построение полной и точной диаграммы классов для системы «Интерфейс электронной почты». Это включает:

* Определение основных классов, представляющих роли пользователей (гость, пользователь, администратор) и их функциональность.
* Выделение атрибутов, отражающих данные, связанные с каждым классом, например, учетная информация, права доступа или статистика активности.
* Описание методов, представляющих функционал, такой как отправка писем, регистрация пользователей, управление безопасностью и мониторинг активности.
* Формализацию связей между классами, таких как отношения наследования, ассоциации и композиция, чтобы четко определить взаимодействие между различными частями системы.

Разработка диаграммы классов поможет не только визуализировать логику приложения, но и создать прочную основу для дальнейшей реализации кода. Это позволит выявить возможные проблемы на ранних этапах проектирования, улучшить коммуникацию между разработчиками, а также обеспечить базу для масштабирования системы в будущем.Описание программных средств.

# Описание программных средств

Для создания моделей в данной работе использовалось программное средство Draw.io, которое является универсальным инструментом для проектирования диаграмм и графических схем. Этот инструмент широко применяется для разработки UML-диаграмм, блок-схем, интеллект-карт, бизнес-процессов, а также схем сущностей и отношений. Благодаря своей функциональности и удобству, Draw.io активно используется как студентами, так и профессиональными разработчиками.

Draw.io представляет собой онлайн-сервис и настольное приложение с открытым исходным кодом, доступное для всех пользователей бесплатно. Это делает программу привлекательной для широкого круга задач, связанных с визуализацией и проектированием. В ней предусмотрен интуитивно понятный интерфейс, который позволяет быстро создавать схемы любой сложности.

Инструмент обладает богатым функционалом, таким как:

* Более 500 шаблонов элементов и фигур.
* Облегчённый интерфейс, в котором за короткий промежуток времени можно создать готовый проект.
* Поддержка горячих клавиш, задействованных в большинстве графических редакторов.
* Экспорт в форматы: JPG, PNG, SVG, VDSX.
* Возможность совместной работы.
* Наличие различных фоновых тем.
* Мультиязычный интерфейс.

Программа поддерживает мультиязычный интерфейс и предоставляет различные темы оформления, что позволяет адаптировать её под личные предпочтения.

Отдельного внимания заслуживает возможность совместной работы, что особенно полезно для командных проектов. Также Draw.io поддерживает импорт файлов из других инструментов, включая Microsoft Visio (.vsdx), Gliffy™ и Lucidchart™, что облегчает работу с уже существующими диаграммами.

Таким образом, Draw.io выступает надежным и функциональным инструментом, обеспечивающим эффективное создание и редактирование диаграмм в рамках проектирования программных систем и других областей.

# Описание практического задания

Для реализации функциональности интерфейса электронной почты были выделены основные сущности приложения: User, Guest, Admin, Contact, Email, Folder.

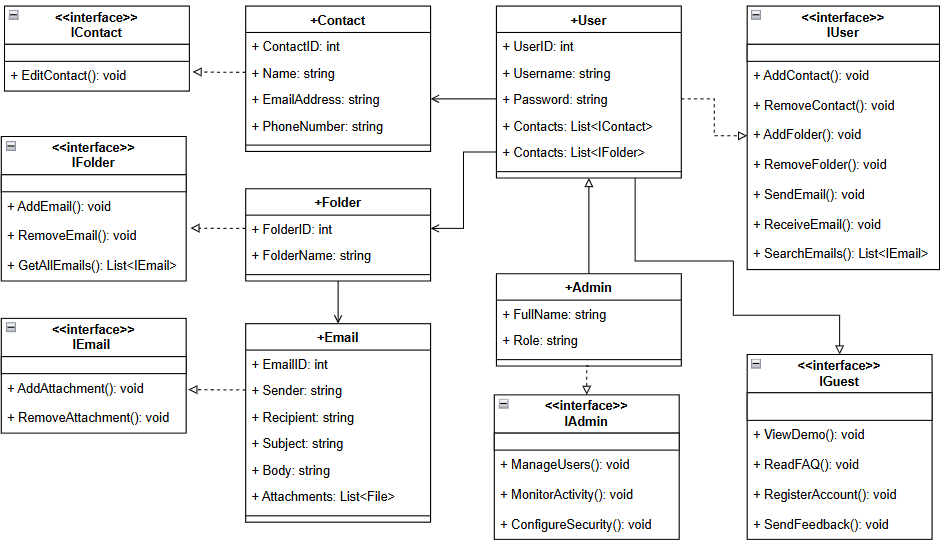


Рисунок 4.1 – Диаграмма классов

Эти классы представляют ключевые сущности, отвечающие за хранение данных и предоставление функциональности через методы. Каждый класс включает набор атрибутов и методов, обеспечивающих их базовое поведение.

Кроме того, в системе используются интерфейсы:

* IUser, который определяет общие действия для всех пользователей.
* IAdmin, описывающий возможности системного администратора.
* IGuest, предоставляющий функционал для гостей системы.

Связи между классами:

* Используется агрегация в отношениях между пользователем и его контактами, папками, а также между папками и письмами. Это позволяет одному объекту (например, папке) содержать ссылки на другие объекты (письма) без жёсткой зависимости.
* Ассоциация применяется для обозначения слабых связей, таких как связь между папками и пользователем, указывая на использование одного объекта другим.

Данная структура упрощает реализацию и расширение функциональности благодаря четкому распределению обязанностей между классами и интерфейсами. Сгруппировав все элементы, можно выделить логические пакеты для удобства работы:

* UserPresentation. Отвечает за представление пользовательского интерфейса для различных ролей: гости, зарегистрированные пользователи, администраторы. Содержит элементы, связанные с отображением пользовательского интерфейса.
* UserDomain. Содержит логику работы пользователей (гости, зарегистрированные, администраторы). Реализует взаимодействие между пользователями и их данными.
* DataMapper. Обеспечивает преобразование данных между доменной моделью и базой данных. Включает классы для сохранения, извлечения и обновления данных пользователей, контактов, писем.
* Database. Отвечает за работу с базой данных.
* EmailPresentation. Реализует логику отображения данных о письмах. Например, интерфейсы для просмотра, поиска и фильтрации писем.
* EmailDomain. Содержит бизнес-логику, связанную с почтой. Обрабатывает отправку, получение писем, архивацию, удаление.
* UIFramework. Обеспечивает общий интерфейс для взаимодействия пользовательских компонентов.

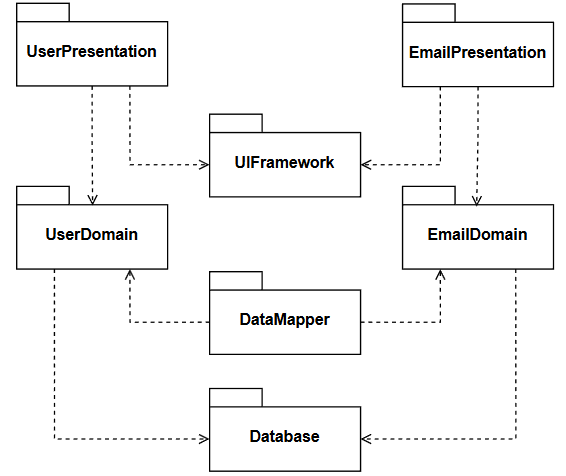


Рисунок 4.2 – Диаграмма пакетов

Связи между пакетами:

* UserPresentation взаимодействует с UserDomain для отображения данных.
* EmailPresentation использует EmailDomain для работы с данными о письмах.
* UserDomain и EmailDomain связаны с DataMapper, который преобразует их данные для Database.
* UIFramework связывает визуальные и логические компоненты, предоставляя общий интерфейс.

Эта структура обеспечивает модульность и четкое разделение обязанностей между пакетами, что делает приложение легко поддерживаемым и расширяемым.