Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работная работа по предмету «Проектирование информационных систем» на тему «Объектно-ориентированное модулирование. Структурные диаграммы UML»

Студент: Ковкель Н. В.

ФИТ 4 курс 4 группа

Преподаватель: Якубенко К. Д.

Минск 2024

# 1 Постановка задач

Задачей проекта является моделирование процессов информационной системы на основе методологии IDEF3, разработка моделей бизнес–процессов и их декомпозиции. Модели должны отражать все функциональные требования, заявленные к информационной системе на основании предыдущих лабораторных работ.

Web–приложение должно быть выполнено с использованием асинхронного программирования, взаимодействовать с базой данных, реализовано под разными платформами. Web–приложение должно представлять собой web–приложение с асинхронным UI с использованием фреймворка React Native. Отображение, бизнес–логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Диаграмму вариантов использования разработать на основе UML, также необходимо разработать логическую схему базы данных и структурную схему приложения. Язык разработки проекта Python, платформа «Django Rest Framework».

Функционально web-приложение должно:

* поддерживать роли гостя, пользователя.

Обеспечивать гостям возможности:

* зарегистрироваться;
* аутентифицироваться;
* авторизоваться.

Обеспечивать пользователям возможности:

* взаимодействовать через свайп-систему с потоком людей;
* ставить лайки;
* пропускать анкеты;
* редактировать профиль (логин, пароль, аватар, предпочтения);
* получать уведомления о новых сообщениях и взаимных симпатиях;
* общаться с другими пользователями через чат.

Основные страницы веб-приложения:

* страница регистрации;
* страница авторизации;
* главная страница;
* личный кабинет пользователя;
* страница уведомлений;
* чат;
* настройки;
* страница помощи и поддержки.

Для гарантированной безопасности пользователей приложение должно применять метод хеширования паролей SHA256 перед их сохранением в базу данных.

# 2 Описание программных средств

Draw.io является мощным инструментом для создания диаграмм и визуального моделирования, который использовался для разработки и документирования архитектуры системы в процессе выполнения лабораторной работы. Приложение предоставило интуитивно понятный графический интерфейс для создания структурных схем, которые помогли визуализировать ключевые компоненты системы, а также их взаимосвязи и процессы взаимодействия. Использование draw.io стало важным шагом для проектирования системы, так как оно упростило процесс создания диаграмм, описывающих функциональные блоки приложения и их взаимодействие. Приложение также поддерживает интеграцию с различными облачными сервисами, что облегчило совместную работу над схемами и предоставило гибкость в управлении проектами.

Инструмент draw.io был выбран за его следующие особенности:

* Доступность: кроссплатформенная, поддержка работы в браузере, а также на Windows, macOS и Linux.
* Поддержка стандартов: возможность создания UML–диаграмм, блок–схем, диаграмм ER, IDEF0 и IDEF3, что позволило эффективно визуализировать бизнес–процессы и архитектуру системы.
* Легкость в использовании: интуитивно понятный интерфейс, который позволяет создавать диаграммы без необходимости владения сложными графическими навыками.
* Интеграция с облачными хранилищами: поддержка Google Drive, OneDrive, GitHub, что упрощает доступ к схемам и их совместное редактирование.

Основное назначение draw.io в этом проекте заключалось в создании визуальных моделей бизнес–процессов и архитектурных решений, которые легли в основу разработки системы. Эти диаграммы помогли в структурировании процессов разработки и визуализации потоков данных, что обеспечило более четкое понимание системы в целом.

Данные о технологии draw.io:

* Разработчик: JGraph Ltd.
* Адрес загрузки: [app.diagrams.net](https://app.diagrams.net).
* Использовался для: создания диаграмм и визуальных моделей бизнес–процессов и архитектурных решений системы.
* Доступность: кроссплатформенная, поддержка веб–версии и настольных приложений.

# 3 Описание практического задания

При разработке системы взаимодействуют несколько ключевых объектов, которые обеспечивают выполнение функциональных требований. Эти объекты включают в себя такие сущности, как пользователи, роли, сообщения, чаты, файлы и другие, каждая из которых определена специфическими атрибутами. Эти сущности связаны между собой, формируя структуру и поведение системы.

Диаграмма классов позволяет описать данные сущности. Она представлена на рисунке 3.1.

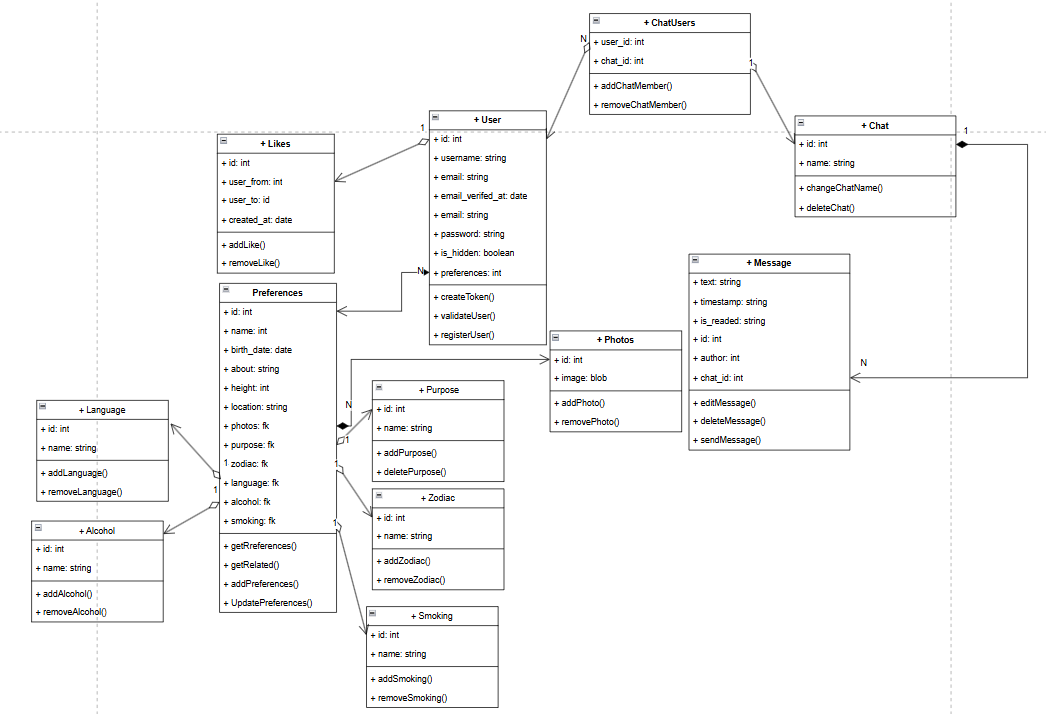


Рисунок 3.1 – Диаграмма классов

Модель информационной системы мобильного приложения для знакомств «EventFlow» была спроектирована на основе диаграммы, которая демонстрирует ключевые сущности и их взаимосвязи. Центральной сущностью является «User» (Пользователь), которая включает в себя такие атрибуты, как имя пользователя, электронная почта, дата верификации электронной почты и настройки видимости. Эта сущность служит основой для всех взаимодействий в системе, обеспечивая возможность каждому пользователю настроить свои параметры и взаимодействовать с другими участниками приложения.

Каждый пользователь может настроить свои «Preferences» (Предпочтения), которые включают персональные данные: имя, дату рождения, рост, информацию о предпочтениях (цели использования приложения, знак зодиака, наличие детей, языки общения, отношение к курению и алкоголю). Эти параметры помогают лучше адаптировать приложение к потребностям пользователя, улучшая поиск и взаимодействие с другими участниками. Сущности, такие как «Purpose», «Zodiac», «Children», «Language», «Alcohol», «Smoking», и «Sex», детализируют различные аспекты предпочтений пользователя, позволяя персонализировать взаимодействие внутри системы.Помимо этих основных сущностей, система поддерживает сущность «**Файл»** для управления вложениями. Файлы имеют такие свойства, как имя файла, путь и тип, которые могут включать изображения или аудиофайлы, прикрепленные к сообщениям. Связь между файлами и сообщениями определена таким образом, что каждое сообщение может ссылаться на файл при необходимости, либо же на видео или фотографию .

Кроме того, приложение поддерживает механизмы социального взаимодействия через сущности «Likes» (Лайки) и «Subscription» (Подписки). Эти сущности фиксируют действия пользователя по отношению к другим: кто и кому поставил лайк, а также подписался на обновления. Система хранит информацию о времени создания таких взаимодействий, что позволяет отслеживать историю активностей пользователей.

Для обеспечения общения между пользователями система использует сущность «Chat» (Чат), которая поддерживает групповые и приватные беседы. Каждый чат связан с несколькими пользователями, что реализовано через связь «многие ко многим». Внутри чатов пользователи обмениваются сообщениями, которые представлены сущностью «Message» (Сообщение). Сообщения включают такие атрибуты, как текст, временная метка, статус прочтения, а также автор и чат, в котором они были отправлены. Это обеспечивает гибкий и удобный механизм для коммуникаций между пользователями. Так же структура таблицы позволяет в случае необходимости добавить еще виды сообщения такие как аудио, файлы и видео.

Также приложение поддерживает работу с медиафайлами, предоставляя пользователям возможность загружать и управлять фотографиями через сущность «Photo» (Фото), которая связана с предпочтениями пользователя.

Кроме того, приложение содержит дополнительные элементы для управления поведением системы, такие как пользовательские компоненты для логирования и обработки ошибок. Диаграмма пакетов показана на рисунке 3.2.

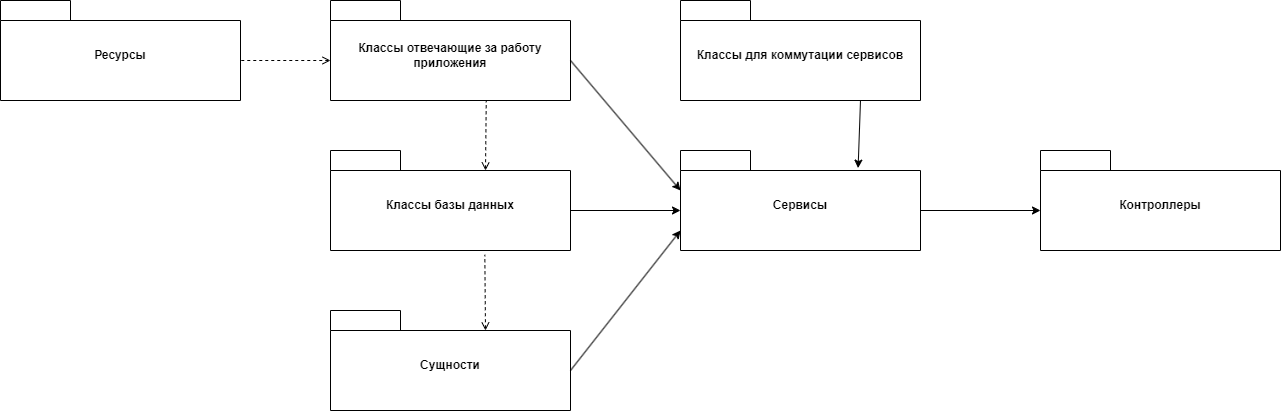


Рисунок 3.2 – Диаграмма пакетов

Кроме того, диаграмма пакетов служит инструментом для планирования и документирования системы, что особенно важно при работе в команде. С её помощью можно эффективно обсуждать архитектурные решения и изменения, обеспечивая всеобъемлющее понимание текущей структуры данных. Это позволяет избежать дублирования усилий и улучшает координацию между различными группами разработки.

Также диаграмма может использоваться для оценки влияния изменений в одном пакете на другие пакеты, что особенно актуально в условиях быстрой итерации и непрерывной интеграции. Понимание взаимосвязей и зависимостей помогает выявлять риски и своевременно их минимизировать.

В дальнейшем, наличие четкой и наглядной диаграммы пакетов может облегчить процесс обучения новых участников команды, позволяя им быстрее вникнуть в архитектуру системы и её логику. Это в свою очередь повышает общую продуктивность команды и способствует созданию более устойчивой и масштабируемой архитектуры.

.

# 4 Ответы на вопросы

1) Перечислите структурные диаграммы, которые входят в UML 2.0.

В UML 2.0 введено 13 типов диаграмм, разделенных на два подмножества – структурных и поведенческих диаграмм.

Структурные модели, известные также как статические модели, описывают структуру сущностей или компонентов некоторой системы, включая их классы, интерфейсы, атрибуты и отношения. К структурным относятся следующие диаграммы:

* диаграммы пакетов или контейнеров (Package diagrams);
* диаграммы классов (Class or Structural diagrams);
* диаграммы объектов (Object diagrams);
* композитные диаграммы (Composite Structure);
* диаграммы компонентов (Component diagrams), показывающие образование структур из классов и интерфейсы между структурами;
* диаграммы развертывания (Deployment diagrams).

2) Укажите назначение структурных диаграммы.

Основное назначение структурных диаграмм заключается в графическом представлении состава статистических совокупностей, характеризующихся как соотношение различных частей каждой из совокупностей. Состав статистической совокупности графически может быть представлен с помощью как абсолютных, так и относительных показателей.

3) Опишите нотации, которые используются для построения Classes диаграмм.

Существует несколько обозначений диаграмм классов, которые используются при рисовании диаграмм классов UML. Мы перечислили ниже наиболее распространенные нотации диаграммы классов.

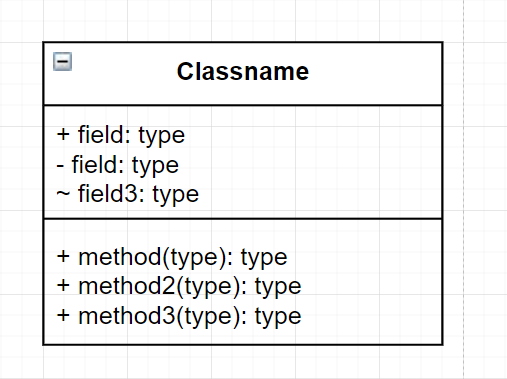


Рисунок 4.1 – Изображение класса

Классы представляют собой центральные объекты в системе. Он представлен прямоугольником с 3 отсеками.

Первый показывает имя класса, а средний – атрибуты класса, которые являются характеристиками объектов. В нижнем списке перечислены операции класса, которые представляют собой поведение класса.

Последние два отсека являются необязательными. Нотация класса без последних двух отделений называется простым классом и содержит только имя класса.

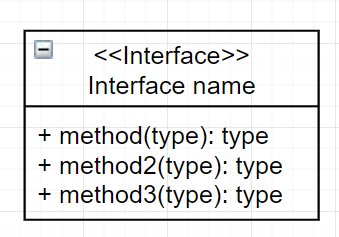


Рисунок 4.2 – Изображение интерфейса

Символ интерфейса на диаграммах классов обозначает набор операций, которые детализируют ответственность класса.

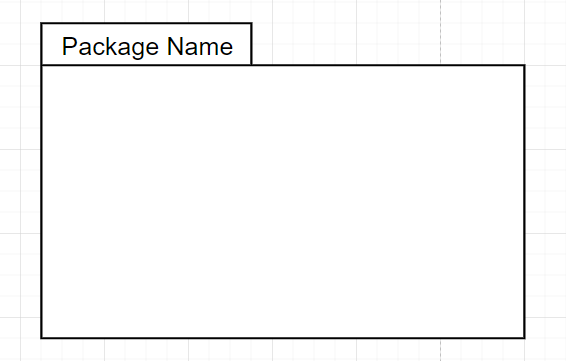


Рисунок 4.3 – Изображение пакета

Символ пакета используется для группировки классов или интерфейсов, которые либо похожи по своей природе, либо связаны. Группировка этих элементов дизайна с использованием символов упаковки улучшает читабельность диаграммы.

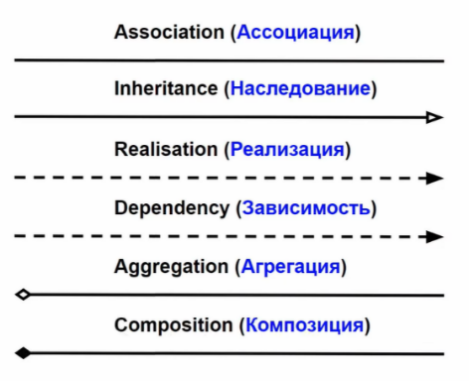


Рисунок 6 – Отношения в диаграмме классов

4) Для чего применяются расширения диаграмм UML?

Механизмы расширения UML включают: стереотипы (stereotype) - расширяют словарь UML, позволяя на основе существующих элементов языка создавать новые, ориентированные для решения конкретной проблемы; помеченные значения (tagged value) - расширяют свойства основных конструкций UML, позволяя включать дополнительную информацию в спецификацию элемента.

5) Что означают понятия «стереотип»?

Стереотип**-**класс, связанный с "метамоделью", что означает стандартные элементы UML, определенные в стандарте UML. Таким образом, это класс классов (или класс ассоциаций, или любого другого элемента UML).