Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа № 5.

Объектно-ориентированное моделирование. Структурные диаграммы UML.

Студент: Яшный Н. С.

ФИТ 4 курс 4 группа

Преподаватель: Якубенко К. Д.

Минск 2024

1. **Тема и цель работы**

Темой данной лабораторной работы является построение UML-диаграммы вариантов использования, которая выступает важным инструментом для графического представления взаимосвязей между ролями конечных пользователей и вариантами взаимодействия с системой. В данном случае речь идет о сервисе поиска недвижимости «HomeHub», который был представлен в первой лабораторной работе. Этот сервис предоставляет пользователям возможность находить, сравнивать и управлять объектами недвижимости, включая квартиры, дома и коммерческую недвижимость. В условиях современного рынка, где информационные технологии играют ключевую роль, такой сервис становится особенно актуальным и востребованным, обеспечивая удобство и доступность информации для пользователей.

Целью лабораторной работы является глубокое изучение методологии объектно-ориентированного моделирования с использованием средств UML. Этот процесс включает в себя не только знакомство с основными принципами объектно-ориентированного проектирования программного обеспечения, но и практическое применение этих принципов при создании диаграмм. Важно отметить, что UML предоставляет стандартизированный язык для описания различных аспектов программной системы, что способствует более эффективному общению между членами команды разработки. В процессе работы студенты получат необходимые навыки для проектирования функциональности информационной системы, что является критически важным для успешного выполнения задач, связанных с разработкой программного обеспечения.

Изучение UML-диаграмм вариантов использования позволяет не только лучше понять, как конечные пользователи взаимодействуют с системой, но и выявить потребности различных групп пользователей. Это знание помогает определить, какие функции они могут выполнять и какие требования предъявляются к разработчикам для реализации этих функций. Кроме того, графическое представление взаимодействий создает общее представление о системе, что упрощает дальнейшее тестирование и поддержку. Это делает UML-диаграммы неотъемлемой частью современного подхода к разработке программного обеспечения, обеспечивая структурированный и организованный процесс проектирования.

Таким образом, данная лабораторная работа направлена на формирование у студентов практических и теоретических навыков, необходимых для качественного и эффективного проектирования информационных систем. Она также способствует углублению понимания объектно-ориентированного подхода в программировании и важности системного мышления при разработке сложных программных решений. Студенты смогут перенести полученные знания на реальные проекты, что значительно повысит их профессиональную подготовку в сфере разработки программного обеспечения.

1. **Описание функциональных требований**

Функциональные требования к системе можно разделить на требования функционалу для различных ролей приложения – пользователя, администратора и гостя.

Функционал для пользователя:

* обеспечивать возможность регистрации, авторизации и аутентификации;
* просмотр каталога недвижимости;
* поиск и фильтрация недвижимости;
* просмотр подробной информации о недвижимости;
* просмотр информации о продавце;
* добавление объявлений в избранное;
* публикация собственных объявлений;
* удаление собственных объявлений;
* просмотр новостей;
* возможность оставить комментарий;
* просмотр геолокации недвижимости на карте.

Функционал для администратора:

* просмотр списка всех пользователей;
* публикация новостей;
* возможность изменить роль пользователя;
* возможность заблокировать/разблокировать пользователя;
* удаление любого объявления;
* просмотр геолокации недвижимости на карте.

Функционал для гостя:

* регистрация;
* просмотр каталога недвижимости;
* поиск и фильтрация недвижимости;
* просмотр подробной информации о недвижимости;
* просмотр информации о продавце;
* просмотр новостей;
* просмотр геолокации недвижимости на карте.

1. **Описание программных средств**

Для построения UML-диаграмм вариантов использования использовался веб-ресурс Draw.io, разработанный компанией JGraph Ltd. Этот инструмент предназначен для создания различных типов диаграмм и визуализаций, что делает его незаменимым помощником для разработчиков, проектировщиков и менеджеров. Адрес веб-ресурса – <https://www.drawio.com>.

Draw.io доступен на всех платформах, которые имеют веб-браузер и доступ в Интернет, что обеспечивает его универсальность и удобство использования. Пользователи могут работать с ним на различных устройствах, включая настольные компьютеры, ноутбуки и даже мобильные устройства, что делает процесс создания диаграмм еще более гибким и доступным. Также стоит отметить, что Draw.io не требует регистрации или создания учетной записи, что упрощает доступ к инструменту и позволяет быстро начать работу.

Функционал веб-ресурса Draw.io весьма разнообразен и включает в себя возможность построения не только UML-диаграмм, но и других графических элементов, таких как графики, смысловые карты, диаграммы Венна, Agile и Kanban доски, а также графики мозговых штурмов и диаграммы архитектур технических систем. Это позволяет пользователям создавать визуализации, которые соответствуют самым различным потребностям и задачам, будь то разработка программного обеспечения, проектирование бизнес-процессов или образовательные цели. Например, возможность создания диаграмм рабочего процесса может быть полезной для оптимизации бизнес-операций, а диаграммы архитектуры могут помочь в представлении сложных технических систем.

Кроме того, Draw.io предлагает интеграцию с популярными облачными хранилищами и платформами, такими как Google Drive, OneDrive, GitHub и многие другие. Эта функциональность позволяет пользователям легко сохранять свои работы и делиться ими с коллегами, что особенно важно для командной работы и совместного проектирования. Интеграция с такими платформами также обеспечивает возможность совместного редактирования в реальном времени, что значительно улучшает взаимодействие между членами команды.

Таким образом, использование Draw.io для создания UML-диаграмм вариантов использования не только упрощает процесс визуализации, но и значительно увеличивает эффективность работы команды, позволяя сосредоточиться на ключевых аспектах проектирования и разработки. Это делает инструмент идеальным выбором для организаций, стремящихся к повышению производительности и качеству своей работы в области разработки программного обеспечения и управления проектами.

1. **Описание практического задания**

В ходе выполнения практического задания необходимо построить диаграмму классов. Данная диаграмма описывает типы объектов системы и различного рода статические отношения, которые существуют между ними.

В каждом прямоугольнике, отражающем класс, содержатся имя класса, его атрибуты (поля и свойства) и операции (методы).

Связи разделяются на зависимость (если изменения одного вызовут изменения другого), ассоциацию (в случае, когда один класс пользуется функционалом другого), обобщения (наследование всех открытых полей, свойств и методов), агрегации (включение в другой класс в виде составной части, при которой зависимый класс не может существовать без основного) и композиции (также включение в виде составной части другого класса, но составной класс не может существовать обособленно от основного).

Диаграмма классов для сервиса «HomeHub» представлена на рисунке 4.1.



Рисунок 4.1 – Диаграмма классов сервиса «HomeHub»

Рассмотрим подробнее составные части данной программы. В ней содержится 8 классов со следующими атрибутами и операциями:

* Класс: User. Атрибуты: id, firstName, lastName, login, password, active, phoneNumber, role. Операции: GetOneUser, Login, Register, Auth, Users, BanUser, ChangeUserRole;
* Класс Realt. Атрибуты: id, name, price, type, roomsCount, dealType, country, city, street, house, article, area, images, previewImageId, dateOfCreated, user. Операции: AddImageToRealt, OnCreate, GetRealts, GetRealt, CreateRealt, DeleteRealt;
* Класс Image. Атрибуты: id, name, originalFileName, size, contentType, isPreviewImage, bytes, realt. Операции: GetImageById.
* Класс DealType. Атрибуты: id, dealTypeName. Операции: GetDealTypes, AddDealType, DeleteDealType.
* Класс Type. Атрибуты: id, typeName. Операции: GetTypes, AddType, DeleteType.
* Класс Favorite. Атрибуты: id, user, realt. Операции: GetFavorites, AddFavorite, DeleteFavorite.
* Класс Post. Атрибуты: id, title, anons, full\_text, comments, dateOfCreated. Операции: OnCreate, GetPosts, GetPost, AddPost, DeletePost.
* Класс Comment. Атрибуты: id, text, user, post, dateOfCreated. Операции: OnCreate, GetComments, AddComment, DeleteComment.

Данные сущности имеют следующие связи:

* Пользователи User связаны с недвижимостью Realt агрегационной связью один-ко-многим, причем у пользователя может быть 0 или несколько объектов недвижимости.
* Пользователи User связаны с комментариями Comment композиционной связью один-ко-многим, причем у пользователя может быть 0 или несколько комментариев.
* Пользователи User связаны с избранным Favorites композиционной связью один-ко-многим, при у пользователя может быть 0 или несколько избранных.
* Недвижимость Realt связана с изображениями Image композиционной связью один-ко-многим, причем у недвижимости может быть 1 или несколько изображений.
* Недвижимость Realt связана с типами сделки DealType агрегационной связью один-ко-многим, при у недвижимости может быть 1 тип сделки, но к одному типу сделки может относится много объектов недвижимости.
* Недвижимость Realt связана с типами Type агрегационной связью один-ко-многим, причем у недвижимости может быть 1 тип, но к одному типу может относится много объектов недвижимости.
* Недвижимость Realt связана с избранным Favorites композиционной связью один-ко-многим, причем недвижимость может быть в 0 или нескольких избранных.
* Новости Post связаны с комментариями Comment композиционной связью один-ко-многис, причем у меня новости может быть 0 или несколько комментариев.

На рисунке 4.2. представлена диаграмма пакетов приложения «HomeHub»

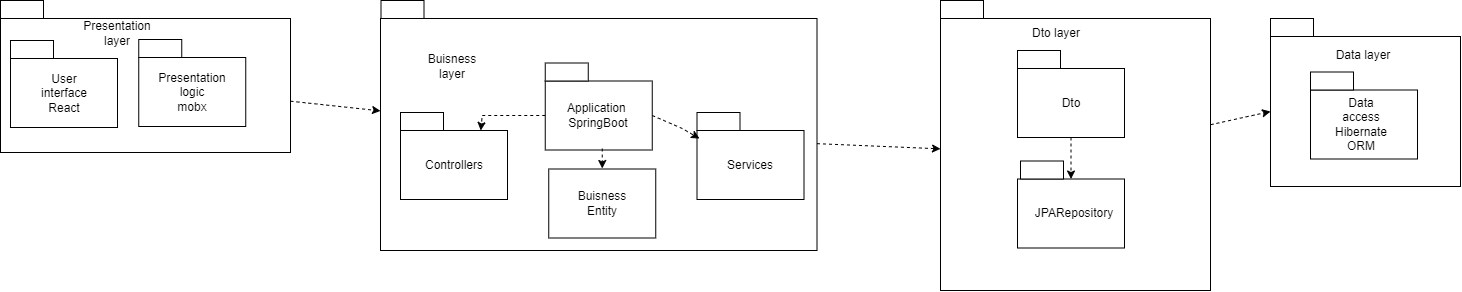


Рисунок 4.2. – диаграмма пакетов приложения «HomeHub»

На диаграмме видно, Presentation layer состоит из React и mobx, Business layer состоит из SpringBoot с контроллерами, сервисами и сущностями, Dto layer состоит из Dto и JPARepository, Data layer состоит из Hibernate ORM.

1. **Ответы на вопросы**

Следующие ответы на вопросы:

* Структурные диаграммы в UML 2.0: диаграмма классов, диаграмма компонентов, диаграмма развертывания, диаграмма объектов, диаграмма пакетов, диаграмма структур.
* Назначение структурных диаграмм. Структурные диаграммы предназначены для отображения статической структуры системы. Они помогают визуализировать: классы и их атрибуты, связи между классами, компоненты системы и их взаимодействия, организацию системы в пакеты или модули.
* Нотации для построения диаграмм классов. Основные нотации для диаграмм классов включают: классы: прямоугольники, разделённые на три части (имя, атрибуты, методы), Связи: Линии, соединяющие классы, с различными обозначениями (ассоциации, агрегации, композиций), модификаторы доступа: символы + (публичный), - (приватный), # (защищённый) перед атрибутами и методами, интерфейсы: Овал или прямоугольник с «<<interface>>», представляющий контракт.
* Применение расширений диаграмм UML. Расширения диаграмм UML используются для добавления специфической информации или метаданных, которые не охватываются стандартными нотациями. Это позволяет: уточнять определение элементов модели, интегрировать дополнительные функции или концепции, специфичные для проекта, совместить UML с другими методологиями и фреймворками.
* Стереотип и тегированное значение. Стереотип: это расширение, которое позволяет классифицировать элементы модели. Например, можно обозначить класс как «<<entity>>», «<<control>>» или «<<boundary>>», чтобы указать его роль в системе. Тегированное значение: это пара "ключ-значение", ассоциированная с элементом модели, которая предоставляет дополнительную информацию. Например, можно указать «{version=1.0}» для класса, чтобы обозначить версию.