МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт КНТ

Факультет ИСП

Кафедра ПИ

Научно-исследовательская работа

на тему: «Разработка графовой базы данных и средств взаимодействия с ней для работы сервиса доставки продуктов питания»

Руководитель Выполнил

Морозова О.В. ст. гр. ПИ-18в

Жильцов В.А.

Донецк – 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc132412873)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc132412874)

[2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ 8](#_Toc132412875)

[3 ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 10](#_Toc132412876)

[3.1 Проектирование концептуальной модели БД 10](#_Toc132412877)

[3.2 Теоритическая основа для реализации микросервиса для взаимодействия с базой данных 15](#_Toc132412878)

[3.3 Практическая реализации микросервиса для взаимодействия с базой данных 16](#_Toc132412879)

[3.4 Реализация клиентской части приложения 19](#_Toc132412880)

[3.5 Реализация серверной части приложения 19](#_Toc132412881)

[3.6 Реализация мобильного приложения 19](#_Toc132412882)

[ЗАКЛЮЧЕНИЕ 20](#_Toc132412883)

[ПЕРЕЧЕНЬ ССЫЛОК 21](#_Toc132412884)

# ВВЕДЕНИЕ

В настоящее время успешное функционирование различных фирм, организаций и предприятий просто невозможно без развитой информационной системы, которая позволяет автоматизировать сбор и обработку данных. Одним из самых удобных способов взаимодействия с пользователями является веб-сайт, который должен соответствовать современным тенденциям, а именно быть адаптивным, быстрым и приятным.

Прежде чем начать разработку, необходимо изучить предметную область фирмы, для которой создается продукт, это может помочь при выборе используемого инструментария и паттернов проектирования. Под предметной областью принято понимать некоторую область человеческой деятельности или область реального мира, подлежащих изучению для организации управления и автоматизации, например, предприятие, интернет-магазин, сервис доставки продуктов питания и т.д.

В условиях интенсивной цифровизации экономики и появления новых трендов потребительских предпочтений на российском рынках все большую популярность приобретают новые формы онлайн-торговли продуктами питания и готовой едой, которые в рамках указанных трендов стали практически неотделимы друг от друга.

Пандемия коронавируса резко ускорила процесс перехода к формату онлайн-торговли продуктами питания (ОТПП). Согласно исследованию, проведенному Аналитическим центром НАФИ в апреле 2020 года, большинство российских интернет-пользователей (67%) за время самоизоляции совершали покупки онлайн, а каждый четвертый (26%) заказывал доставку продуктов питания на дом. Начали пользоваться услугами доставки продуктов питания во время самоизоляции 13% россиян, столько же (13%) указали, что пользовались услугой доставки продуктов и ранее[1]. По оценке M.A. Research, в 2020 году российский объем ОТПП вырос до 174 млрд рублей, что составило 1% от всего рынка продовольственного ритейла[2]. В 2021—2025 годах совокупный среднегодовой темп роста ОТПП будет иметь долю, по разным оценкам, от 33 до 40%. В 2022 году сегмент ОТПП вырастет до 415—445 млрд рублей при выполнении заявленных планов компаний, а его объем в обороте розничной торговли продуктами питания достигнет 2,2—2,4%[3]. Действительно, спрос на интернет-услугу доставки продуктов питания в целом по России продолжает расти, так, за 6 месяцев 2021 года он вырос на 73% по сравнению с тем же периодом 2020 года[4].

Данный тренд особенно характерен для крупных городов: например, в Санкт-Петербурге за первые полгода после введения пандемийных ограничений количество онлайн-заказов продуктов на дом увеличилось в 20 раз[5].

Учитывая процесс трансформации и увеличения объема рынка ОТПП можно говорить о том, что разработка интернет-магазинов, а также сервисов, связанных с ними, для компаний является актуальной и востребованной сферой деятельности.

Предметом исследования является ОТПП с использованием современных технологий создания, сбора статистики и администрирования интернет-магазинов.

Цель научно-исследовательской работы – разработка автоматизированной информационной системы «Сервис доставки продуктов питания».

Для достижения указанной цели в выпускной квалификационной работе выполняются следующие задачи:

* изучение и применение на практике технических аспектов разработки сервисов;
* проведение анализа и выбор технологий для разработки приложений сервиса;
* детали разработки приложений, входящих в состав сервиса доставки еды.

Выпускная квалификационная работа состоит из введения, одного раздела, заключения и одного приложения.

# 1 ПОСТАНОВКА ЗАДАЧИ

Разрабатываемая система должна хранить подробную информацию о заказах, клиентах, сотрудниках компании, кухнях, а также предоставлять удобные средства просмотра статистики. Исходя из этого, были составлены следующие требования:

1. Сервис должен предоставлять клиенту удобную возможность просмотреть список доступных блюд, заказать их, видеть состояние заказа и местонахождение курьера, а также зарегистрироваться или авторизоваться.
2. Разработанный сайт должен быть адаптивным, удобным, а также быстрым, что может быть достигнуто за счет использования таких технологий как Server-side Generation, обеспечивающих предварительное создание статических файлов в момент сборки.
3. Сервис должен предоставлять администраторам такие статистические данные по заказам как:

* скорость перехода заказа из состояния «В очереди» в состояние «Готовится»;
* скорость приготовления заказа;
* скорость передачи заказа доставщику;
* скорость доставки;
* соотношение завершенных заказов и отмененных.

1. Сервис должен обладать эффективными системами аутенфикации и авторизации, позовляющим определить роль пользователя. Отдельно стоит заметить, что пользователь не должен каждый раз при входе в систему проходить систему аутентификации.
2. Сервис должен предоставлять отдельный функционал сотрудником кухонь, которые должны видеть выполняемые заказы и текущую очередь заказов.

Исходя из составленных требований системы, были выделены четыре роли пользователей: администратор, сотрудник кухни, курьер, а также клиент.

Основные возможности администратора включают в себя:

* управление заказами;
* управление кухнями;
* просмотр общей статистики;
* управление блюдами.

Основные возможности сотрудника кухни включают в себя:

* управление некоторыми состояниями заказа;
* просмотр очереди заказов;
* просмотр статистики кухни.

Основные возможности курьра включают в себя:

* управление некоторыми состояниями заказа;
* просмотр очереди заказов, а также истории выполненных заказов.

Основные возможности клиента включают в себя:

* просмотр блюд;
* просмотр истории собственных заказов;
* просмотр корзины;
* оформление заказа;
* просмотр профиля.

Исходя из вышеперечисленного можно понять, что разрабатываемый сервис должен состоять из нескольких программ.

Мобильное приложение для курьеров, которое будет отслеживать их местоположение и предоставлять им все данные, необходимые для работы. Стоит заметить, что приложение будет нацелено на пользователей смартфонов с оперционной системой Android.

Интернет-сайт, который будет предоставлять удобный функционал для клиентов, сотрудников кухонь, а также администраторов.

Серверная часть должна предоставлять API для взаимодействия с графовой базой данных как для сайта, так и для мобильного приложения.

# 2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ

Тщательно изучив предметную область, было решено использовать графовую базу данных для хранения данных.

Графовые базы данных (Graph databases) - это тип баз данных, который использует графы для моделирования и хранения данных. Графовые базы данных хранят данные в виде узлов и ребер, которые представляют объекты и связи между ними.

Преимущества графовых баз данных:

* гибкость моделирования данных: графовые базы данных могут хранить данные различных типов, что делает их идеальным инструментом для моделирования сложных структур данных;
* скорость запросов: графовые базы данных используют оптимизированные алгоритмы для обработки запросов, что позволяет осуществлять поиск данных в режиме реального времени и получать результаты быстрее, чем с традиционными реляционными базами данных;
* масштабируемость: графовые базы данных могут быть легко масштабированы по мере увеличения объема данных.

Недостатками графовых баз данных являются:

* ограничения в масштабируемости: хотя графовые базы данных могут быть масштабируемыми, это может быть сложно и затратно в настройке для больших объемов данных;
* ограничения по производительности: хотя графовые базы данных могут обеспечивать высокую производительность для некоторых типов запросов, они могут быть медленнее для других типов запросов, таких как запросы, связанные с агрегированием данных;
* сложность моделирования данных: хотя графовые базы данных могут быть гибкими в моделировании данных, это может также стать причиной сложностей в проектировании базы данных и требовать более тщательного планирования структуры данных.

Одной из наиболее популярных графовых баз данных является Neo4j, которая будет использоваться в проекте.

Для разработки клиентской части приложения, помимо стандартных HTML, CSS, JavaScript, будет использоваться библиотека Bootstrap, для удобной работы со стилями, а также React – библиотека для JavaScript, предназначенная для работы со страницей, как с набором компонентов, что увеличивает скорость обновления и загрузки страницы.

Для серверной части будет использоваться язык C#, а именно фреймворк ASP.NET Core. ASP.NET Core - фреймворк для разработки веб-приложений, который предоставляет широкий спектр функциональных возможностей для разработки масштабируемых и безопасных веб-приложений. Наиболее характерными его достоинствами являются:

* высокая производительность за счет оптимизации запросов и использованию многопоточности;
* безопасность, которая обеспечивается путем предоставления встроенных механизмов для защиты от таких угроз, как атаки инъекций, кросс-сайт скрипты и многих других.
* простота разработки.

Для мобильного приложения будет использоваться язык программирования Kotlin, который создан на основе Java Virtual Machine и разрабатывался, чтобы упростить и ускорить процесс разработки для Android. Kotlin обладает понятным и лаконичным синтаксисом, совместимостью с Java, следовательно, он может использоваться все библиотеки, которые были написаны на Java, а также высокой производительностью.

# 3 ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## 3.1 Проектирование концептуальной модели БД

После тщательного анализа предметной области была разработана база данных, состоящая из 9 узлов, а также 8 связями.

Было создано 9 типов узлов:

1. “Order” – заказ;
2. “Admin” – администратор;
3. “Client” – клиент;
4. “DeliveryMan” – курьер;
5. “KitchenWorker” – сотрудник кухни;
6. “Kitchen” – кухня;
7. “Dish” – блюдо;
8. “OrderState” – состояние заказа;
9. “Category” – категория блюда.

Было создано 8 типов связей между узлами:

1. “Ordered” – оформленый заказ клиентом;
2. “ReviewedBy” – оставленный отзыв на заказ клиентом;
3. “DeliveredBy” – доставленный заказ курьером;
4. “WorkedIn” – сотрудник кухни, работающий на кухне;
5. “CookedBy” – заказ, приготовленный кухней;
6. “HasOrderState” – заказ, имеющий состояние;
7. “ContainDish” – категория, содержащая блюдо;
8. “OrderedDish” – блюдо, находящееся в заказе;

Исходя из имеющихся данных, была создана графовая модель данных (см. рис. 3.1)

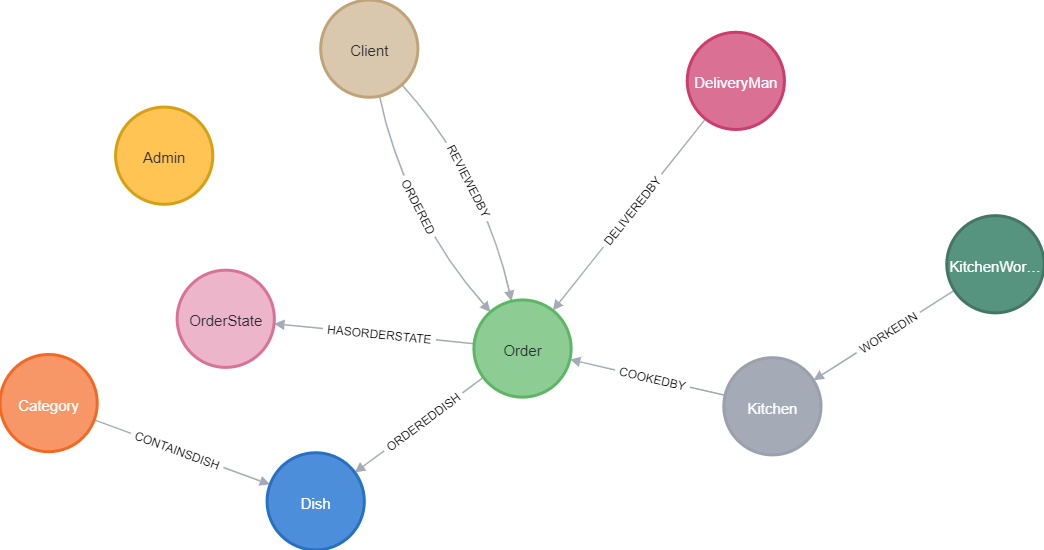


Рисунок 3.1 – Схема графовой базы данных «Сервис доставки продуктов питания»

Параметры узлов приведены в таблицах 3.1 – 3.9.

Таблица 3.1 – «Order»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | DeliveryAddress | Адрес доставки |
| 3 | Price | Стоимость заказа |
| 4 | StoryJson | История изменения состояний в формате json |
| 5 | SumWeight | Суммарный вес продуктов |

Таблица 3.2 – «Kitchen»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | Address | Адрес кухни |

Таблица 3.3 – «Admin»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | Born | Дата рождения |
| 3 | Login | Логин |
| 4 | Name | Имя |
| 5 | PasswordHash | Хеш пароля |
| 6 | PhoneNumber | Номер телефона |

Таблица 3.4 – «DeliveryMan»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | Born | Дата рождения |
| 3 | Login | Логин |
| 4 | Name | Имя |
| 5 | PasswordHash | Хеш пароля |
| 6 | PhoneNumber | Номер телефона |
| 7 | MaxWeight | Максимальный вес, который может переносить курьер |

Таблица 3.5 – «OrderState»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | DescriptionForClient | Описание состояния заказа для клиента |
| 3 | NameOfState | Название состояния |
| 4 | NumberOfStage | Номер состояния |

Таблица 3.6 – «KitchenWorker»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | Born | Дата рождения |
| 3 | Login | Логин |
| 4 | Name | Имя |
| 5 | PasswordHash | Хеш пароля |
| 6 | PhoneNumber | Номер телефона |
| 7 | JobTitle | Должность |

Таблица 3.7 – «Dish»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | Description | Описание блюда |
| 3 | DirectoryWithImages | Папка с изображениями блюда |
| 4 | IsAvailable | Доступен ли клиенту |
| 5 | Name | Название блюда |
| 6 | Price | Стоимость за одну единицу |
| 7 | Weight | Вес |

Таблица 3.8 – «Category»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | CategoryNumber | Номер категории |
| 3 | Description | Описание категории |
| 4 | Name | Название категории |

Таблица 3.9 – «Client»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Id | Уникальный идентификатор |
| 2 | Born | Дата рождения |
| 3 | Login | Логин |
| 4 | Name | Имя |
| 5 | PasswordHash | Хеш пароля |
| 6 | PhoneNumber | Номер телефона |
| 7 | Bonuses | Количество бонусов |

Помимо узлов, данные также хранят связи. Всего создано 8 типов связей, каждая из которых хранит свой уникальный идентификатор, уникальный идентификатор начального и конечного узла. Однако 4 связи имеют дополнительные свойства, они продемонстрированы в таблицах 3.10 – 3.13.

Таблица 3.10 – Уникальные свойства связи «WorkedIn»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | GotJob | Дата вступления в должность |

Таблица 3.11 – Уникальные свойства связи «ReviewedBy»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | ClientRating | Оценка пользователя |
| 2 | TimeCreated | Время создания отзыва |
| 3 | Review | Отзыв |

Таблица 3.12 – Уникальные свойства связи «OrderedDish»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | Count | Количество заказанных блюд |

Таблица 3.13 – Уникальные свойства связи «HasOrderState»

|  |  |  |
| --- | --- | --- |
| № | Name | Comment |
| 1 | TimeStartState | Время начала действия состояния |
| 2 | Comment | Комментарий изменения состояния |

Приведенных свойств достаточно для работы с необходимыми данными, а также корректного функционирования системы.

## 3.2 Теоритическая основа для реализации микросервиса для взаимодействия с базой данных

В течении длительного периода времени создавались и оттачивались подходы для взаимодействия с базой данных (БД). Один из таких подходов – Object Relation Mapping.

Object Relation Mapping (ORM) – архитектурный паттерн, который связывает БД с принципами объектно-ориентированных языков программирования. Суть данной технологии заключается в том, что каждая строка реляционной БД сопоставляется с объектом класса, который задает разработчик. В свойство объекта помещаются данные из соответствующего столбца, сопоставление происходит чаще всего по имени свойства и столбца. Благодаря использованию данной технологии мы можем взаимодействовать с каждой строкой БД как с объектом. Если речь идет о графовой БД, то свойства каждого узла и связи между узлами сопоставляются со свойствами объекта.

Многие паттерны способны дополнять друг друга, ORM не является исключением. Данный паттерн является основой для более масштабных шаблонов, самые популярные из них это ActiveRecord, DataMapper, Repository. В современной разработке паттерн Repository стал наиболее популярным, так как он является наиболее гибким и масштабируемым.

Repository – паттерн, представляющий собой уровень доступа к данным, который выполняет двунаправленную передачу данных между постоянным хранилищем данных (зачастую базой данных) и уровнем отображения данных (веб страница, окно приложения и т.д.). Паттерн выполняет функцию изолирования уровня БД и отображения. Слой состоит из одного или нескольких репозиториев (классов манипулирования информации указанного типа), выполняющих передачу данных. Реализации репозиториев могут различаться по объему и предоставляемому функционалу, так универсальные репозитории будут использовать обобщенные типы и тем самым обрабатывать множество различных типов, а специализированные репозитории будут обрабатывать указанные типы.

В самом простом виде, репозиторий будет иметь связь, показанную на рисунке 3.1.

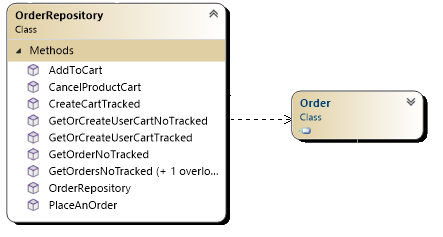


Рисунок 3.2 –Примитивная реализация паттерна Repository

В таком виде паттерн хорошо подходит для реляционных БД, однако графовые БД требуют иной подход, так как они могут хранить данные не только в узлах, но и в связях.

## 3.3 Практическая реализации микросервиса для взаимодействия с базой данных

Предложенное развитие паттерна Repository состоит с следующем. Для начала необходимо определить интерфейс IModel, который будут расширять все остальные интерфейсы и реализовывать классы узлом и связей базы данных. Все классы, реализующие данный интерфейс, будут обязаны определить свойства и методы, находящиеся в данном интерфейсе. Одним из таких свойств является свойство Id, которое служит уникальным идентификатором.

Интерфейс IModel расширяют интерфейсы INode и IRelation, которые предназначены для узлов и связей соответственно.

Класс Node реализует интерфейс INode. Данный класс выступает базовым для узлов, хранящихся в БД. Интерфейс IRelation реализует обобщенный класс Relation <TNodeFrom, TNodeTo>, они являются базовыми для связей в БД. Помимо уникального идентификатора, интерфейс будет задавать ссылки на узлы, между которыми будет проложена связь, а также сами узлы. Не всегда возникает необходимость указывать из какого в какой узел идет связь, однако в текущем проекте использовать база данных Neo4j, которая создает исключительно направленные связи.

Реализованные интерфейсы и классы изображены на рисунке 3.3.

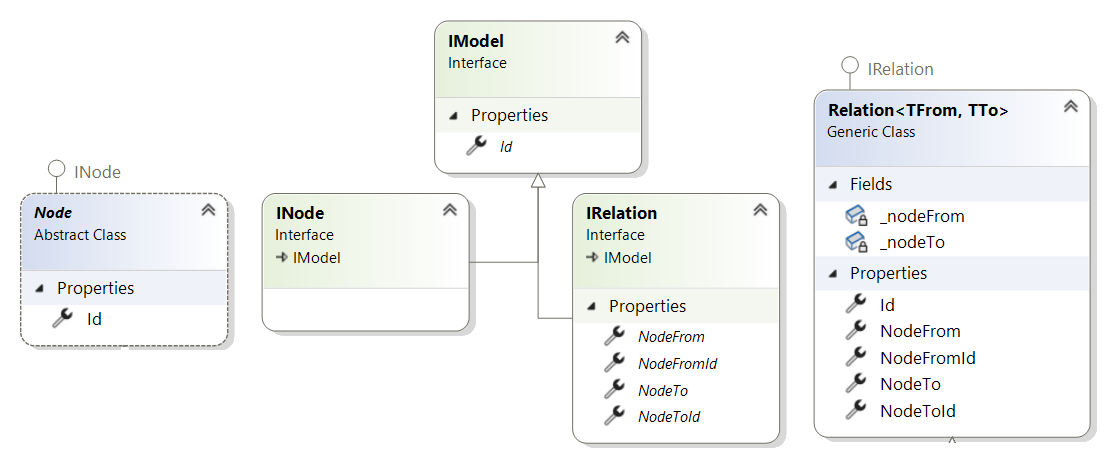


Рисунок 3.3 – Диаграмма базовых классов для данных, хранимых в БД

Следующим шагом будет создание непосредственно репозиториев и необходимой инфраструктуры подходящих для графовых БД.

Интерфейс IGeneralRepository<TNode> отвечает за определение методов, которые должны быть в каждой реализации универсального репозитория, а класс GeneralRepository<TNode> реализует данный интерфейс. В универсальном классе реализованы основные методы для работы с узлами, но наибольшего внимания требуют методы для работы со связями. Так, если нам нужны связанные узлы с неким определенным типом связи, то нам необходимо указать тип связи, для этого вновь используются обобщения.

На рисунке 3.4 для примера изображена сигнатура метода GetRelatedNodesAsync из интерфейса IGeneralRepository<TNode>.

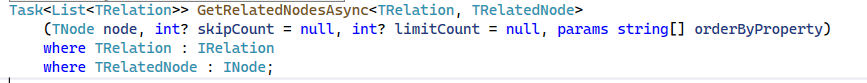
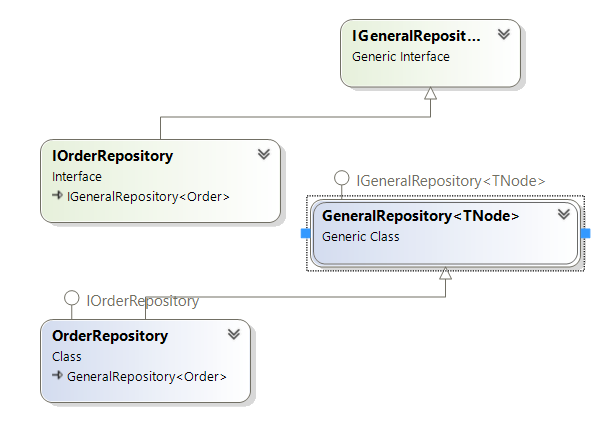


Рисунок 3.4 – Диаграмма базовых классов для данных, хранимых в БД

Данный метод возвращает список связей указанного типа, количество связей будет равняться количеству связанных элементов.

Для получения результата для данного запроса, необходимо определить к какому узлу относится переданный нам объект, он является начальной или финальной точкой связи, для этого необходимо использовать рефлексию. Однако пользователю данного класса нет необходимости об этом беспокоиться.

Разработчику могут понадобиться методы, которые выходят за рамки реализованного функционала, для этого он может создать специализированные репозитории, которые должны наследоваться от универсального. В диаграмме классов на рисунке 3.5 можно увидеть пример реализации специализированного репозитория.

  
Рисунок 3.5 – Диаграмма классов паттерна Repository

## 3.4 Реализация клиентской части приложения

## 3.5 Реализация серверной части приложения

## 3.6 Реализация мобильного приложения

# ЗАКЛЮЧЕНИЕ

# ПЕРЕЧЕНЬ ССЫЛОК