МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт
информационных технологий

Кафедра
информационных систем

КУРСОВОЙ ПРОЕКТ

по дисциплине «**Проектирование информационных систем**»

на тему: «Драйвер для взаимодействия высокоуровневых ЯП с базой данных»

Студент **Музафаров К.Р.**
группа ИДБ–21-06

_____
подпись

Руководитель **Гальчич М.А.**
старший преподаватель

_____
подпись

Москва, 2024 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Тема взаимодействия высокоуровневых языков программирования с базой данных является крайне актуальной в контексте разработки современных информационных систем. С увеличением объемов данных и сложности бизнес-логики становится необходимым эффективное взаимодействие между приложениями, написанными на различных языках программирования, и базами данных.

При этом важно правильно проектировать структуру и взаимодействие компонентов системы, чтобы обеспечить эффективное использование ресурсов и обеспечить безопасность хранения и передачи данных. UML-диаграммы играют ключевую роль в проектировании таких систем, позволяя анализировать и описывать структуру, взаимодействие и поведение компонентов.

Таким образом, изучение и анализ темы взаимодействия высокоуровневых языков программирования с базой данных имеет большое значение для разработчиков программного обеспечения, поскольку позволяет создавать более надежные, эффективные и безопасные информационные системы.

Данный курсовой проект посвящен анализу и проектированию системы взаимодействия высокоуровневых языков программирования с базой данных. В рамках работы были представлены и проанализированы различные виды UML-диаграмм, такие как диаграммы вариантов использования, классов, взаимодействия, размещения, состояний и компонентов.

Проект включает все этапы проектирования, начиная от анализа требований и заканчивая построением детальных диаграмм, описывающих взаимодействие пользователей с системой. Каждый этап анализа и проектирования подробно рассматривался с точки зрения обеспечения качества и надежности программного обеспечения.

Результатом выполнения данного курсового проекта стала разработанная система, удовлетворяющая всем заявленным требованиям, с четкой структурой и функциональностью. Этот проект демонстрирует, как использование DFD, IDEF0, IDEF3, UML-диаграмм и методик объектно-ориентированного анализа и проектирования помогает создать качественное и надежное программное обеспечение.

Анализ диаграмм и описание сценариев взаимодействия позволяют лучше понять процессы работы системы и ее компонентов при обработке данных в контексте высокоуровневых языков программирования.

ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Драйверы играют решающую роль в облегчении взаимодействия между оборудованием и программным обеспечением в компьютерной системе. В частности, драйвер базы данных упрощает обмен данными между клиентским приложением базы данных и сервером базы данных локально или по сети.

Для эффективного взаимодействия с абстрактной базой данных, предназначенной для абстрактного языка программирования, необходимо правильно настроенное программное обеспечение, способное обеспечить надежную и быструю передачу данных между приложением и хранилищем информации. В этом процессе ключевую роль играет драйвер базы данных – программный модуль, обеспечивающий связь между приложением и самой базой данных.

Драйвер для взаимодействия с базой данных выполняет ряд важных функций, таких как установление соединения с абстрактной базой данных, отправка запросов на извлечение или изменение данных, получение ответов от базы данных и их передача обратно в приложение. Каждая операция требует точной обработки и оптимизации для обеспечения стабильной работы приложения и эффективного использования ресурсов абстрактной базы данных.

Драйверы играют решающую роль в облегчении взаимодействия между оборудованием и программным обеспечением в компьютерной системе. В частности, драйвер базы данных упрощает обмен данными между клиентским приложением базы данных и сервером базы данных локально или по сети.

Для эффективного взаимодействия с абстрактной базой данных, предназначенной для абстрактного языка программирования, необходимо правильно настроенное программное обеспечение, способное обеспечить надежную и быструю передачу данных между приложением и хранилищем информации. В этом процессе ключевую роль играет драйвер базы данных –

программный модуль, обеспечивающий связь между приложением и самой базой данных.

Драйвер для взаимодействия с базой данных выполняет ряд важных функций, таких как установление соединения с абстрактной базой данных, отправка запросов на извлечение или изменение данных, получение ответов от базы данных и их передача обратно в приложение. Каждая операция требует точной обработки и оптимизации для обеспечения стабильной работы приложения и эффективного использования ресурсов абстрактной базы данных.

В данном исследовании мы рассмотрим основные принципы работы драйвера для взаимодействия с абстрактной базой данных, его роль в архитектуре приложений и методы оптимизации процесса передачи данных. Понимание работы драйвера базы данных позволит повысить производительность приложений, обеспечить безопасность данных и улучшить общее качество работы с абстрактными базами данных.

Важно отметить, что выбор правильного драйвера базы данных имеет большое значение для производительности и безопасности приложения. Различные базы данных могут требовать разные драйверы, и выбор несовместимого драйвера может привести к снижению производительности или даже к возникновению ошибок. Поэтому разработчики должны тщательно подбирать соответствующий драйвер для конкретной базы данных и языка программирования.

Еще одним важным аспектом является обеспечение безопасности данных при передаче между приложением и базой данных. Драйвер базы данных должен обеспечивать защиту информации от несанкционированного доступа или изменения. Это включает в себя шифрование данных, аутентификацию пользователей и другие меры безопасности, чтобы гарантировать целостность и конфиденциальность информации.

Другим важным аспектом работы драйвера базы данных является оптимизация запросов к базе данных. Хорошо спроектированный драйвер должен уметь эффективно формировать запросы и обрабатывать ответы, чтобы минимизировать нагрузку на сервер базы данных и сократить время ожидания результатов запросов. Такие оптимизации могут значительно повлиять на производительность приложения и удовлетворение пользователей.

Все эти аспекты демонстрируют важность роли драйвера базы данных в разработке приложений. Понимание его функций и возможностей позволяет разработчикам создавать надежные, производительные и безопасные приложения, способные эффективно взаимодействовать с базами данных и обеспечивать высокое качество обработки информации.

ГЛАВА 2. МОДЕЛИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРУКТУРНОГО ПОДХОДА

Моделирование играет ключевую роль в современной науке и технике, позволяя анализировать, предсказывать и оптимизировать различные системы. Структурный подход к моделированию уделяет особое внимание описанию объектов и их взаимосвязей, что позволяет более глубоко понять функционирование систем.

Использование структурного подхода при моделировании обеспечивает более наглядное представление сложных систем, упрощает анализ и улучшает возможности принятия решений. Этот подход также способствует созданию более надежных и эффективных моделей.

Кроме того, структурный подход обеспечивает гибкость при внесении изменений и обновлений в процесс проверки. По мере появления новых факторов или взаимосвязей их можно легко интегрировать в существующую модель, обеспечивая ее актуальность и эффективность с течением времени. Кроме того, иерархический характер модели позволяет выявлять потенциальные узкие места или уязвимости в процессе проверки, позволяя принимать упреждающие меры для их устранения.

В целом, принятие структурного подхода обеспечивает надежную основу для моделирования и анализа процесса проверки контрагентов на надежность, позволяя организациям принимать обоснованные решения и эффективно снижать риски.

Дополнительно, структурный подход позволяет разделить сложные системы на более простые компоненты, что упрощает анализ и понимание работы каждой отдельной части. Это помогает выявить зависимости и влияние одной части системы на другую, что может быть полезно при принятии решений и оптимизации процессов. Также, использование структурного подхода в моделировании способствует более систематическому и организованному подходу к работе с данными и информацией, что повышает точность и достоверность получаемых результатов.

2.1. МЕТОДОЛОГИЯ DFD

DFD — общепринятое сокращение от англ. data flow diagrams — диаграммы потоков данных. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе

источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

Диаграмма потоков данных (data flow diagram, DFD) — один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML. Несмотря на имеющее место в современных условиях смещение акцентов от структурного к объектно-ориентированному подходу к анализу и проектированию систем, «старинные» структурные нотации по-прежнему широко и эффективно используются как в бизнес-анализе, так и в анализе информационных систем.

Исторически сложилось так, что для описания диаграмм DFD используются две нотации — Йордана (Yourdon) и Гейна-Сарсон (Gane-Sarson), отличающиеся синтаксисом.

Информационная система принимает извне потоки данных. Для обозначения элементов среды функционирования системы используется понятие внешней сущности. Внутри системы существуют процессы преобразования информации, порождающие новые потоки данных. Потоки данных могут поступать на вход к другим процессам, помещаться (и извлекаться) в накопители данных, передаваться к внешним сущностям.

Модель DFD, как и большинство других структурных моделей — иерархическая модель. Каждый процесс может быть подвергнут декомпозиции, то есть разбиению на структурные составляющие, отношения между которыми в той же нотации могут быть показаны на отдельной диаграмме. Когда достигнута требуемая глубина декомпозиции — процесс нижнего уровня сопровождается мини-спецификацией (текстовым описанием).

Кроме того, нотация DFD поддерживает понятие подсистемы — структурного компонента разрабатываемой системы.

Нотация DFD — удобное средство для формирования контекстной диаграммы, то есть диаграммы, показывающей разрабатываемую АИС в коммуникации с внешней средой. Это — диаграмма верхнего уровня в иерархии диаграмм DFD. Её назначение — ограничить рамки системы, определить, где заканчивается разрабатываемая система и начинается среда. Другие нотации, часто используемые при формировании контекстной диаграммы — диаграмма SADT, Диаграмма вариантов использования.

Диаграммы потоков данных (DFD) представляют собой графическое изображение потоков данных в информационной системе. Они являются важным инструментом при анализе и проектировании систем, позволяя визуализировать поток информации и процессы, связанные с этими данными. Методология DFD (Data Flow Diagrams) является одним из ключевых инструментов структурного анализа и проектирования информационных систем. Она предоставляет эффективный способ визуализации потоков данных и процессов в системе, позволяя анализировать и проектировать ее на различных уровнях детализации.

Методология базируется на представлении системы в виде совокупности процессов, внешних сущностей и потоков данных, которые между ними циркулируют. Она позволяет абстрагироваться от конкретных деталей и сосредоточиться на ключевых аспектах функционирования системы, что делает ее особенно полезной при анализе сложных процессов, таких как работа драйвера для взаимодействия высокоуровневых ЯП с базой данных.

DFD широко используются для моделирования информационных систем различного масштаба, начиная от небольших приложений до сложных корпоративных систем. Они помогают выявить ключевые процессы и потоки данных в системе, что делает их ценным инструментом анализа и проектирования.

На рисунке 1 представлена контекстная диаграмма уровня системы.

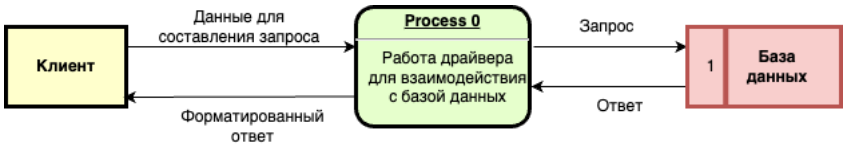

Рис. 1. Диаграмма DFD. Контекстная диаграмма уровня системы.

На рисунке 2 представлена контекстная диаграмма уровня подсистемы.

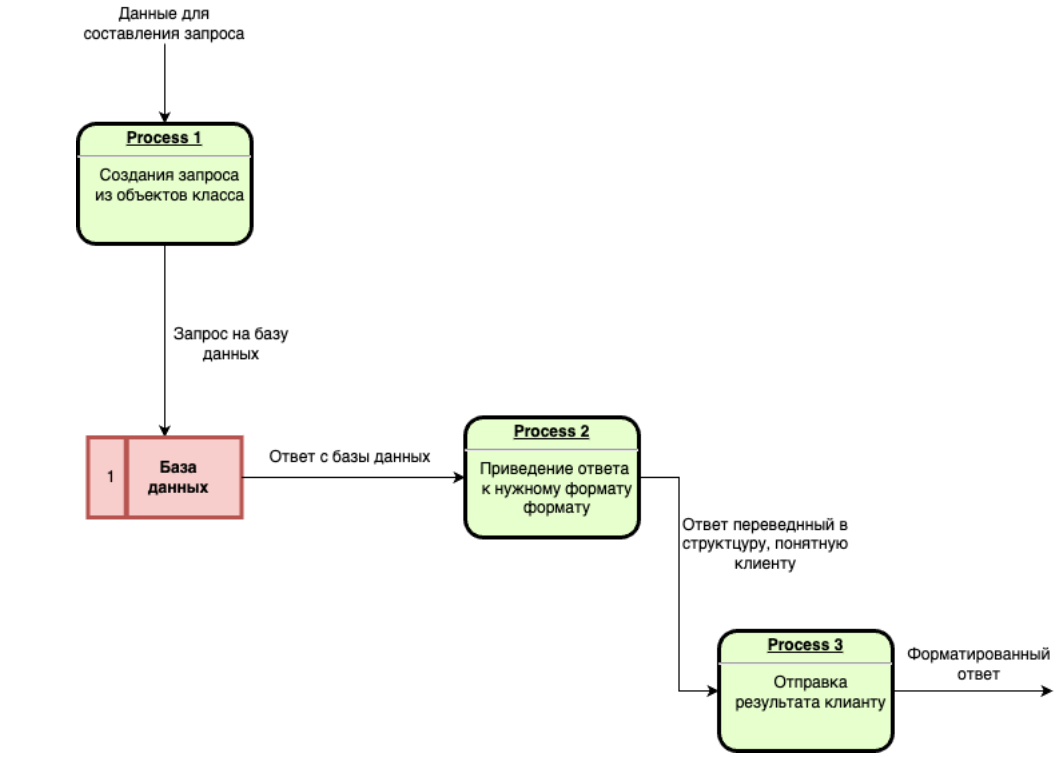

Рис. 2. Диаграмма DFD. Контекстная диаграмма уровня подсистемы.

На рисунке 3 представлена контекстная диаграмма уровня процесса.

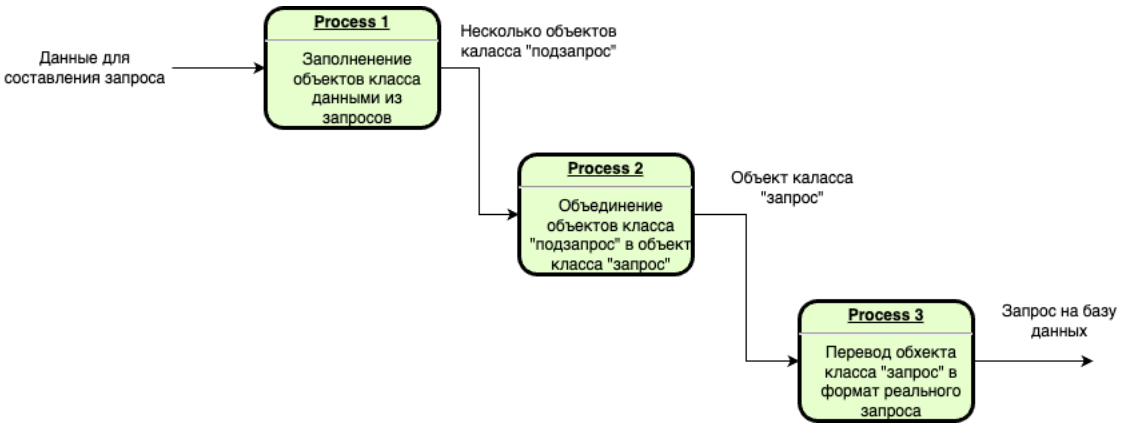

Рис. 3. Диаграмма DFD. Контекстная диаграмма уровня процесса.

2.2. МЕТОДОЛОГИЯ IDEF0

Методология IDEF0 (Integration Definition for Function Modeling) является методом функционального моделирования, используемым для анализа и документирования функций систем. Она позволяет описывать, анализировать и представлять сложные системы в виде функциональных моделей. Методология IDEF0 помогает выявить и описать ключевые функции,

входы, выходы и управляющие отношения в системе с использованием графических обозначений. Этот подход широко применяется в области управления проектами, бизнес-процессов, разработки программного обеспечения и системного анализа.

Описание методологии IDEF0 содержится в рекомендациях Р 50.1.028-2001 «Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования».

Также отображаются все сигналы управления, которые на DFD (диаграмме потоков данных) не отображались. Данная модель используется при организации бизнес-процессов и проектов, основанных на моделировании всех процессов: как административных, так и организационных.

IDEF0 как стандарт был разработан в 1981 году департаментом Военно-воздушных сил США в рамках программы автоматизации промышленных предприятий, которая носила обозначение ICAM (Integrated Computer Aided Manufacturing). Набор стандартов IDEF унаследовал своё название от этой программы (IDEF расшифровывается как ICAM Definition). При этом кроме усовершенствованного набора функций для описания бизнес-процессов, одним из требований к новому стандарту было наличие эффективной методологии взаимодействия в рамках «аналитик-специалист». Другими словами, новый метод должен был обеспечить групповую работу над созданием модели, с непосредственным участием всех аналитиков и специалистов, занятых в рамках проекта.

Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того, чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Также отображаются все сигналы управления, которые на DFD (диаграмме потоков данных) не отображались. Данная модель используется при организации бизнес-процессов и проектов, основанных на моделировании всех процессов: как административных, так и организационных.

На рисунке 4 представлена диаграмма IDEF0 уровень A0.

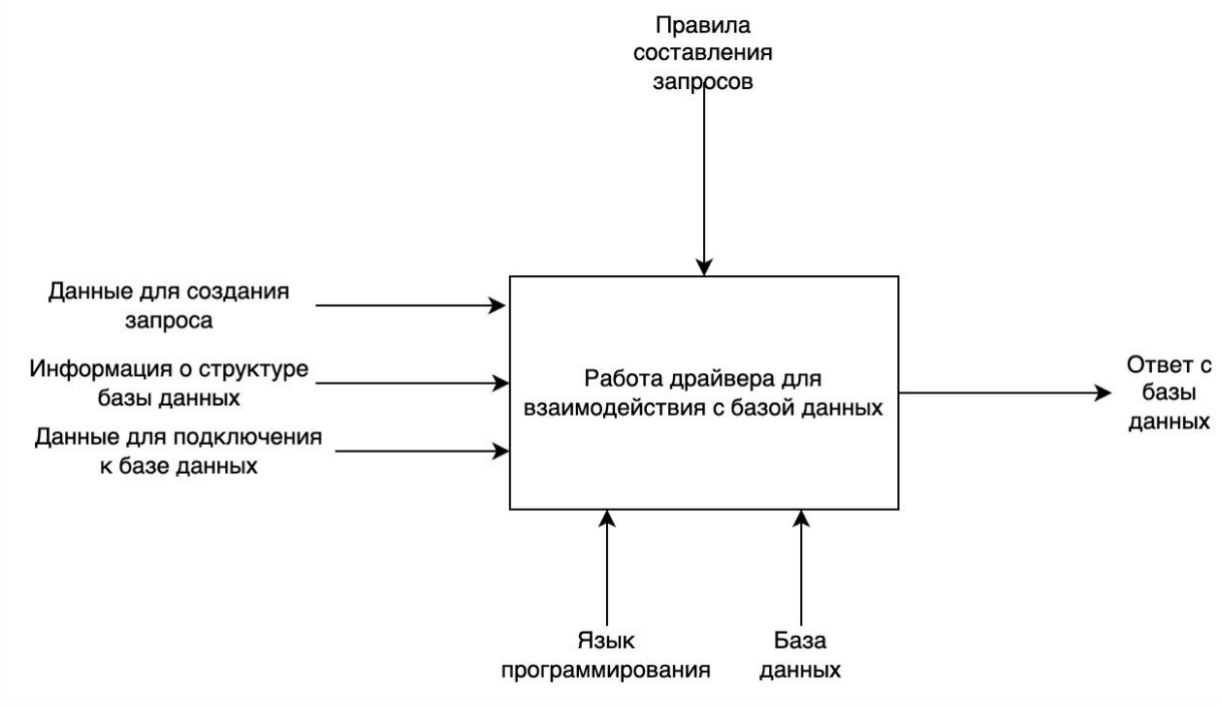

Рис. 4. Диаграмма IDEF0. Уровень A0.

На рисунке 5 представлена диаграмма IDEF0 уровень A1.

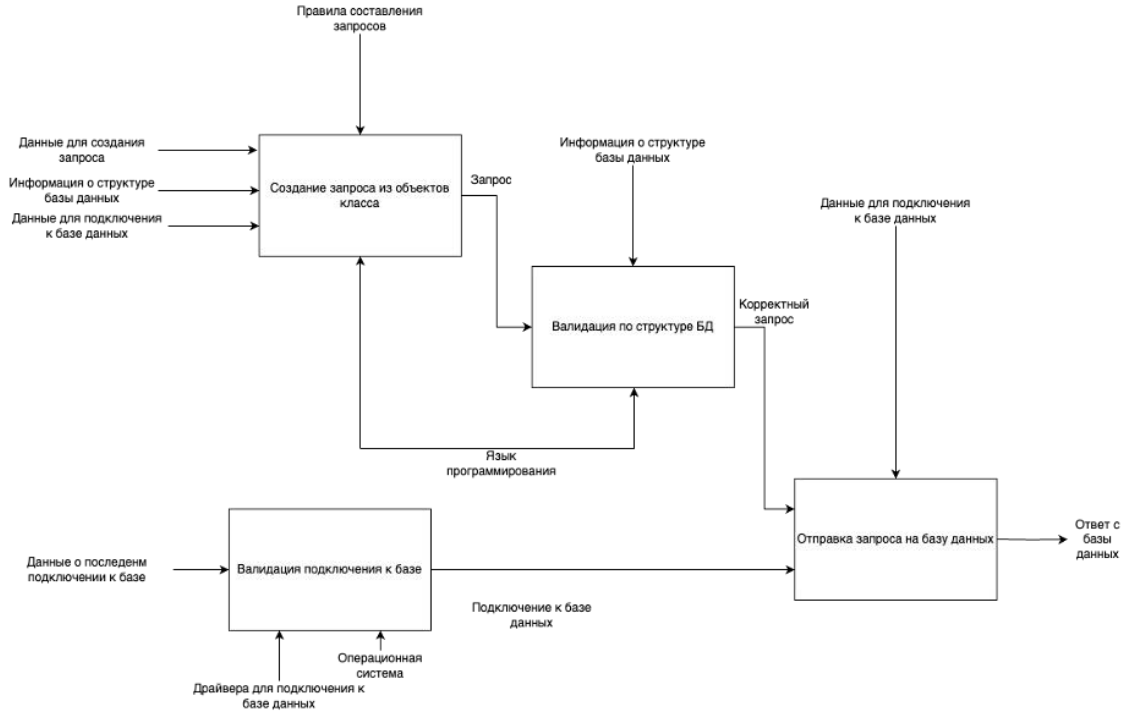

Рис. 5. Диаграмма IDEF0. Уровень A1.

На рисунке 6 представлена диаграмма IDEF0 уровень A2.

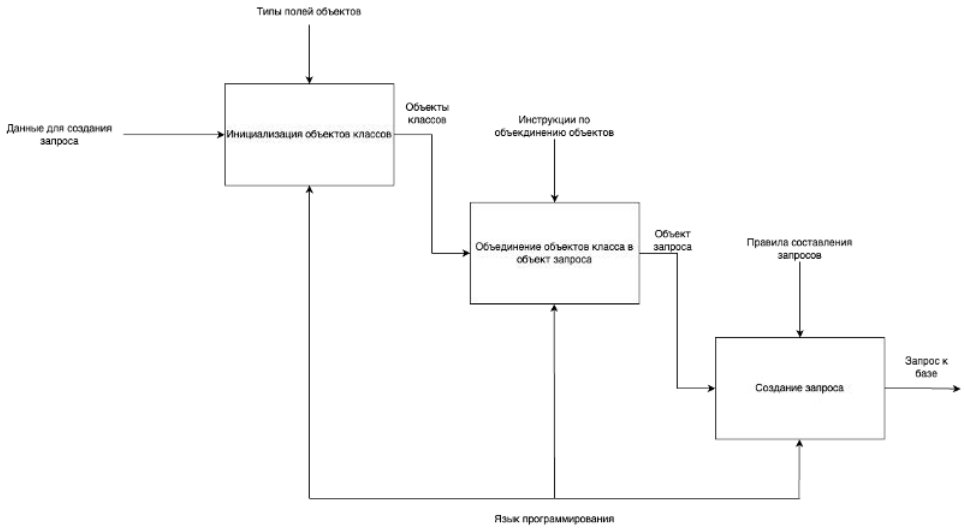

Рис. 6. Диаграмма IDEF0. Уровень A2.

На рисунке 7 представлена диаграмма IDEF0 уровень A3.

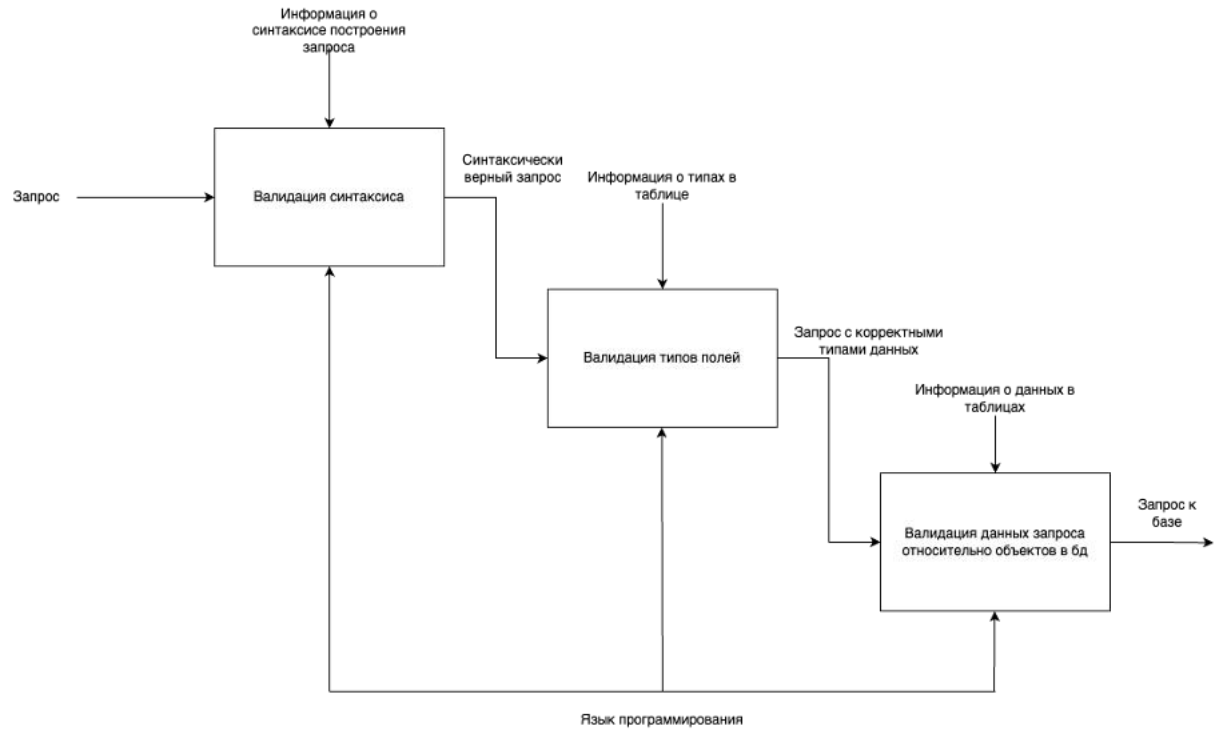

Рис. 7. Диаграмма IDEF0. Уровень A3.

На рисунке 8 представлена диаграмма IDEF0 уровень A4.

Рис. 8. Диаграмма IDEF0. Уровень A4.

2.3. МЕТОДОЛОГИЯ IDEF3

IDEF3 (Integrated DEFinition for Process Description Capture Method) — методология моделирования и стандарт документирования процессов, происходящих в системе. Метод документирования технологических процессов представляет собой механизм документирования и сбора информации о процессах. IDEF3 показывает причинно-следственные связи между ситуациями и событиями в понятной эксперту форме, используя структурный метод выражения знаний о том, как функционирует система, процесс или предприятие.

Моделирование в нотации IDEF3 является частью структурного анализа систем, может использоваться как дополнение и уточнение модели IDEF0.

Система (не обязательно информационная) описывается как упорядоченная последовательность событий с одновременным описанием объектов, имеющих отношение к моделируемому процессу.

Моделирование IDEF3 может быть реализовано двумя альтернативными методами:

Process Flow Description (PFD) — Описание технологических процессов, с указанием того, что происходит на каждом этапе технологического процесса.

Object State Transition Description (OSTD) — описание переходов состояний объектов, с указанием того, какие существуют промежуточные состояния у объектов в моделируемой системе.

Основу методологии IDEF3 составляет графический язык описания процессов, поэтому модель в нотации IDEF3 может содержать два типа

диаграмм: диаграмму Описания Последовательности Этапов Процесса (Process Flow Description Diagrams, PFDD) диаграмму Сети Трансформаций Состояния Объекта (Object State Transition Network, OSTN)

Следует уточнить, что нотация IDEF3 чаще применяется для моделирования и анализа процессов нижнего уровня и может использоваться при декомпозиции блоков процесса модели IDEF0. Однако, сама нотация IDEF3 так же поддерживает возможность декомпозиции, то есть каждый отдельный блок в модели I

На рисунке 9 изображена диаграмма для получения данных с базы.

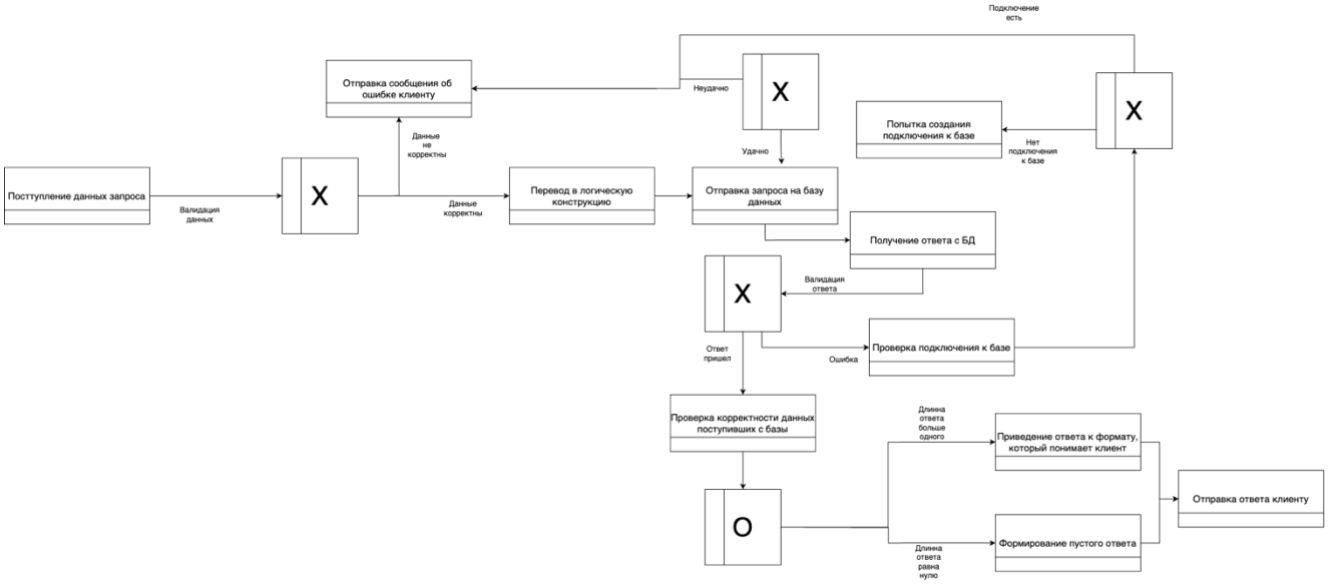

Рис. 9. IDEF3 диаграмма для получения данных с базы.

На рисунке 10 изображена диаграмма для отправки данных на базу.

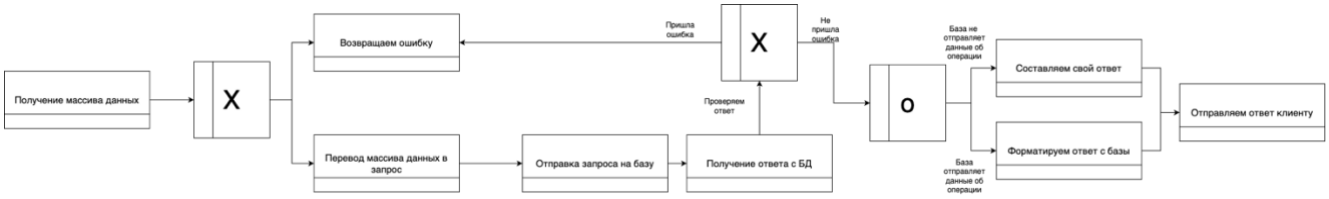

Рис. 10. IDEF3 диаграмма для отправки данных на базу.

На рисунке 11 изображена диаграмма для создания соединения с базой.

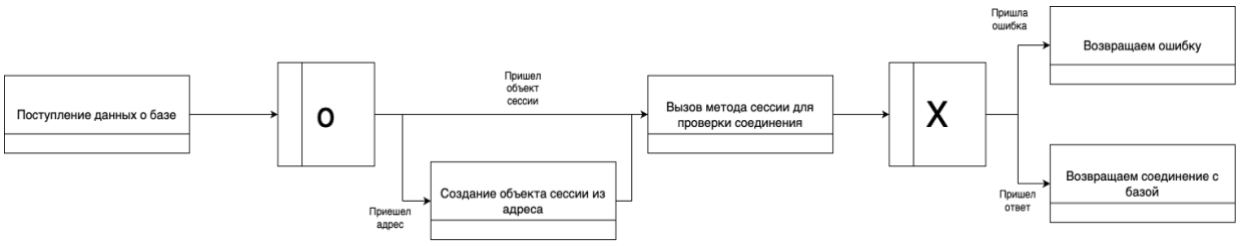

Рис. 11. IDEF3 диаграмма для создания соединения с базой

ГЛАВА 3. МОДЕЛИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА (UML)

Объектно-ориентированное моделирование (ООМ) — это современная технология компьютерного моделирования, широко использующаяся в научных исследованиях, проектировании технических систем и анализе бизнес-процессов и позволяющая создавать надежные компьютерные модели быстрее и дешевле.

UML (Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Язык UML предназначен для разработки спецификаций сложных программноаппаратных систем на ранних стадиях разработки. Его значение для дискретнособытийного моделирования состоит прежде всего в том, что он «канонизировал» сложившиеся в области программирования понятия объектно-ориентированного подхода и, став фактическим стандартом объектно-ориентированного подхода, заставил всех, кто его использует, говорить на одном языке. Кроме того, эти понятия были соединены с исключительно удобными и наглядными диаграммами (машинами) состояний, придуманными еще в 80-х годах прошлого века Д. Харелом. Язык UML легко расширяется для моделирования непрерывно-дискретных (гибридных) систем.

Предпосылки появления языка моделирования UML обозначились в связи с бурным развитием во второй половине XX века объектно-ориентированных языков программирования (Simula 67, Smalltalk, Objective C, C++ и др). Вследствие непрекращающегося усложнения создаваемых программных продуктов возникла нужда в учёте всё новых и новых возможностей языков и средств разработки при анализе, формулировании требований и в процессе проектирования программных приложений. Например, в короткий промежуток времени с 1989 года по 1994 год количество объектно-ориентированных инструментов выросло с десятка до более чем полусотни. Однако многие разработчики затруднялись подобрать язык моделирования, который бы полностью отвечал всем их потребностям. В

результате выделилось новое поколение методов разработки, среди которого особую популярность приобрели метод Буча, созданный Якобсоном Object-Oriented Software Engineering (OOSE) и разработанный Рамбо Object Modeling Technique (OMT). Помимо них существовали и другие завершённые технологии, например Fusion, Shlaer-Mellor и Coad-Yourdon, однако всем из них были присущи не только преимущества, но и существенные недостатки.

Кроме IT, UML используется в проектировании, документировании и построении бизнес-процессов. Он помогает строить схемы, которые визуализируют сложные структуры, действия или понятия. Особенность таких схем в том, что они унифицированы, то есть одинаковые связи и обозначения будут означать одно и то же в разных диаграммах. Это значит в том числе, что любой знающий UML человек легко поймет любую схему, созданную на этом языке.

В целом UML — открытый стандарт, язык широкого профиля. Он нужен для графического описания и визуализации абстрактной модели, а на практике эта модель может быть чем угодно — от архитектуры программы до описания целей деятельности. Название читается как «юмл» или «ю-эм-эл».

Своеобразным индикатором популярности UML может служить тот факт, что в пакете MATLAB появилась новая подсистема StateFlow, поддерживающая диаграммы состояний.

Некоторые основные понятия объектно-ориентированного моделирования и элементы языка UML включают в себя классы и объекты. Классы представляют собой абстрактные шаблоны, определяющие атрибуты и методы объектов, в то время как объекты являются конкретными экземплярами классов, обладающими уникальными значениями атрибутов.

UML предоставляет различные виды диаграмм, такие как диаграммы классов, диаграммы вариантов использования, диаграммы последовательности, диаграммы действий и другие, которые помогают представить различные аспекты системы. UML является стандартом в индустрии разработки программного обеспечения и широко используется разработчиками, аналитиками и дизайнерами для проектирования и документирования систем, обеспечивая общий и понятный язык для коммуникации и совместной работы команд, что приводит к более эффективному процессу разработки программного обеспечения.

3.1. BUSINESS USE CASE DIAGRAMS

Диаграммы использования бизнес-кейсов (Business Use Case diagrams) - это графические представления, которые иллюстрируют взаимодействия между актерами (личностями или системами) и системой или процессом в бизнес-контексте. Они применяются для фиксации, документирования и коммуникации функциональных требований к системе или процессу с точки зрения бизнеса.

Во-первых, диаграммы использования бизнес-кейсов помогают определить и задокументировать функциональные требования, фиксируя взаимодействия между актерами и системой. Они также визуально отображают, как различные актеры взаимодействуют с системой, помогая заинтересованным сторонам понять общий ход действий и процессов.

Кроме того, эти диаграммы служат эффективными инструментами коммуникации между различными участниками проекта, обеспечивая общее понимание функциональности и требований системы. Они помогают определить границы системы и ключевые функциональные возможности.

Кроме того, диаграммы использования бизнес-кейсов могут быть использованы в качестве основы для генерации тестовых случаев, обеспечивая полное покрытие всех функциональных возможностей системы и проверку ожидаемых результатов. Они также помогают выявить и классифицировать различных актеров или сущностей, участвующих в системе, таких как пользователи, внешние системы или другие заинтересованные стороны.

Более того, эти диаграммы часто используются в качестве отправной точки для проектирования архитектуры системы и определения компонентов и интерфейсов. Кроме того, они помогают заинтересованным сторонам выявить области для улучшения, оптимизации рабочих процессов и оптимизации бизнес-процессов, визуально представляя существующие бизнес-процессы и их взаимодействия.

В заключение, диаграммы использования бизнес-кейсов являются ценными инструментами для анализа, проектирования и документирования систем или процессов в контексте бизнеса. Они обеспечивают эффективную коммуникацию, сбор требований и понимание системы среди заинтересованных сторон, а также помогают генерировать тестовые случаи, идентифицировать актеров, проектировать систему и улучшать бизнес-процессы.

Основные бизнес-актеры для драйвера для взаимодействия высокоуровневых ЯП с базой данных:

Заказчик: выдвигает задачу, связанную с данными в каком-либо формате.

Разработчик: пишет программу, которая решает задачу заказчика. Программа при этом взаимодействует с базой данных, чтобы облегчить написанания логики взаимодействия программы с базой данных, и понизить порог вовлеченности разработчика в тематику, другие разработчики написали драйвер для работы с БД.

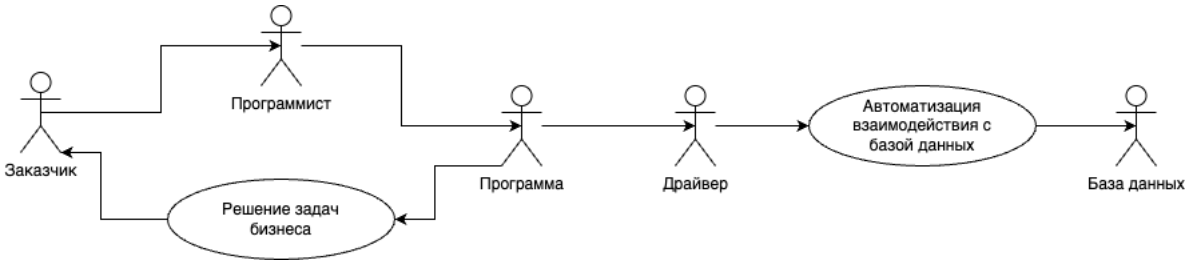

Рис. 12. Business Use Case Diagram.

3.2. ПОСТРОЕНИЕ ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ (USE CASE DIAGRAM)

Диаграмма прецедентов (Use Case Diagram) - по сути, самая распространенная диаграмма, которую можно встретить на просторах сети интернет. Она позволяет визуально изобразить то, какие функции пользователь может выполнять с помощью разрабатываемой системы. И тут самая главная оплошность - нарисовать только диаграмму. Ведь в основе модели прецедентов лежит не столько сама Use Case диаграмма, сколько подробное описание каждого прецедента. Описывая каждый прецедент, аналитик должен расписать основные и альтернативные сценарии его выполнения, а также пред-, постусловия и дополнительные требования. И только тогда, вместе с диаграммой прецедентов, это будет считаться полноценной моделью прецедентов, которая составляется на первых этапах разработки.

На предоставленной диаграмме UML изображены два актора и несколько сценариев взаимодействия с системой. Вот описание актеров и сценариев:

Основные актеры для драйвера для взаимодействия высокоуровневых ЯП с базой данных:

1. Программа
   - Основная роль: Работа с данными, получаемыми из базы.

- Описание: Программа взаимодействует с системой для отправки и получения данных, а также для выполнения операций с базой данных.

2. Система

- Основная роль: Взаимодействие с базой данных.
- Описание: Система осуществляет операции по взаимодействию с базой данных, включая составление запросов, установку соединения и проверку ответов.

Основные сценарии для драйвера для взаимодействия высокоуровневых ЯП с базой данных для "Программа":

1. Отправка данных, которые нужно получить с базы
   - Описание: Программа отправляет запрос на получение определённых данных из базы данных.
2. Отправка данных, которые мы хотим поместить в базу
   - Описание: Программа отправляет запрос на вставку или обновление данных в базе данных.
3. Отправка адреса в базу
   - Описание: Программа отправляет адрес или идентификатор для доступа к конкретным данным в базе данных.
4. Перевод ответа в нужный формат
   - Описание: После получения ответа от базы данных, программа преобразует этот ответ в необходимый формат для дальнейшего использования.

Основные сценарии для драйвера для взаимодействия высокоуровневых ЯП с базой данных для "Система":

1. Составление запросов на получение данных
   - Описание: Система составляет запросы к базе данных для извлечения необходимых данных на основе полученных инструкций.
2. Составление запросов на изменение данных
   - Описание: Система составляет запросы для изменения данных в базе данных (вставка, обновление, удаление).
3. Установка соединения с базой
   - Описание: Система устанавливает соединение с базой данных для выполнения запросов.
4. Проверка ответа
   - Описание: Система проверяет ответы, полученные от базы данных, для подтверждения корректности выполнения запросов и получения данных.

Общие сценарии:

1. Взаимодействие с базой данных

- Акторы: Программа, Система.

- Описание: Оба актора взаимодействуют для выполнения операций по отправке и получению данных, составлению и проверке запросов, установке соединения и обработке ответов от базы данных.

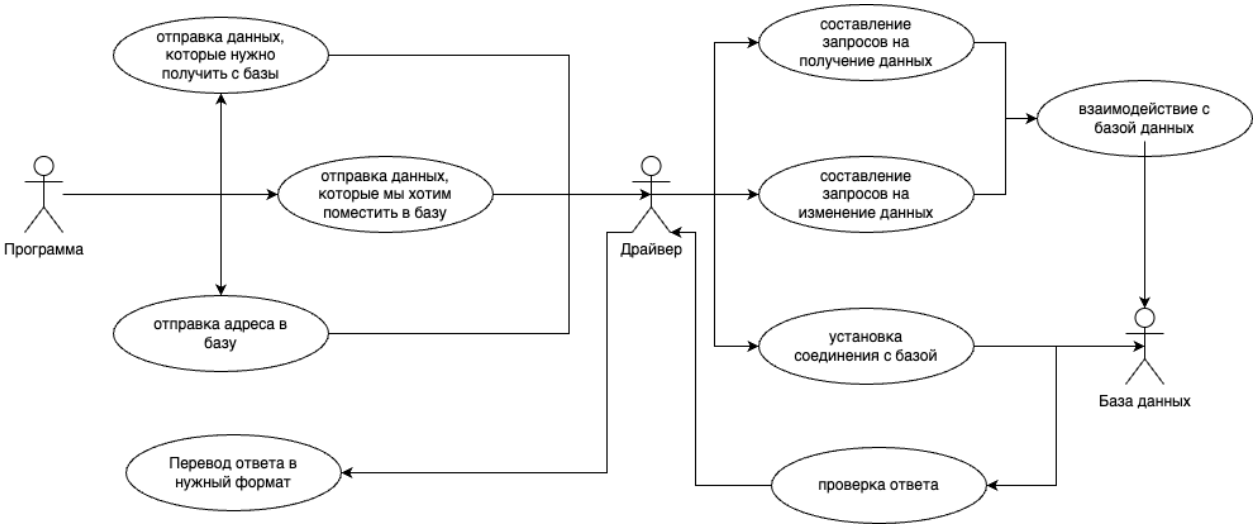

Рис. 13. Use Case Diagram.

ЗАКЛЮЧЕНИЕ

В ходе курсового проекта были представлены и проанализированы диаграммы UML, DFD, IDEF0 и IDEF3, которые позволили описать систему взаимодействия высокоуровневых языков программирования с базой данных. Основная финальная глава проекта содержит подробное описание акторов и сценариев взаимодействия с системой, представленных на диаграмме UML.

В данном курсовом проекте были представлены и проанализированы UML-диаграммы, описывающие систему по поиску контрагентов. Проект включает все этапы проектирования, начиная от анализа требований и заканчивая построением диаграмм.

На этапе анализа требований были определены основные бизнес-актеры и сценарии использования системы. Далее были созданы диаграммы вариантов использования, которые наглядно представляют взаимодействие пользователей с системой.

Спецификация вариантов использования подробно описывает основные и альтернативные потоки каждого сценария, что позволяет точно определить действия пользователей на каждом этапе взаимодействия с системой.

Построение диаграммы классов дало возможность структурировать систему, определить основные классы, их атрибуты и методы, и установить связи между ними.

Диаграммы взаимодействия, такие как диаграммы последовательностей и сотрудничества, показали, как объекты системы взаимодействуют друг с другом в рамках различных сценариев использования.

Диаграмма размещения продемонстрировала физическое распределение компонентов системы, что позволило учесть архитектурные аспекты и обеспечить эффективное функционирование системы в реальных условиях.

Диаграммы состояний и компонентов позволили детально описать жизненный цикл объектов системы и их внутреннюю структуру.

Результатом выполнения курсового проекта стала разработанная система, удовлетворяющая всем заявленным требованиям, с четкой структурой и функциональностью. Эта работа демонстрирует, как использование UML-диаграмм и методик объектно-ориентированного анализа и проектирования помогает создать качественное и надежное программное обеспечение.

Акторы "Программа" и "Система" выступают в качестве основных участников для драйвера, обеспечивающего взаимодействие высокоуровневых ЯП с базой данных. Для "Программы" были описаны сценарии отправки запросов на получение данных, вставку или обновление данных, отправку адреса в базу, а также перевода ответа в нужный формат. Сценарии для "Системы" включают составление запросов на получение и изменение данных, установку соединения с базой и проверку ответов.

Общие сценарии включают взаимодействие обоих акторов для выполнения операций по отправке и получению данных, составлению и проверке запросов, установке соединения и обработке ответов от базы данных.

Таким образом, проведенный анализ диаграмм и описание сценариев взаимодействия позволяют лучше понять процессы работы системы и ее компонентов при обработке данных в контексте высокоуровневых языков программирования.