Содержание

ВВЕДЕНИЕ

Создание программного средства – динамически длительный и трудоемкий процесс. Современные технологии проектирования основаны на последовательной (поэтапной) разработке. По общности целей последовательности работ (этапы) обычно объединяются в стадии.

Технология разработки программного обеспечения (ТРПО) – система инженерных принципов для создания экономичного ПО, которое надежно и эффективно работает в реальных компьютерах. [1]

Курсовой проект по дисциплине ТРПО посвящен разработке программного продукта среднего уровня сложности (размером не менее 1000 операторов универсального языка программирования и с развитым пользовательским интерфейсом).

Цель курсового проекта – приобретение практических навыков в области:

– разработки;

– отладки интерактивных систем программного обеспечения;

– составления пояснительной записки, содержащей обоснование принятых проектных решений;

– применения нормативных документов, регламентирующих состав, содержание и форму технической документации на разработанный программный продукт.

При разработке ИС «картинная галерея» следует учитывать следующие требования:

– стандартизацию основных этапов жизненного цикла программных средств;

– обеспечение надежности и качества функционирования программного средства;

– тестирование нового программного средства.

Приоритетной задачей данного курсового проекта является написание ИС для галереи, но с максимально легкими в обращении интерфейсом и системой управления данными. Это необходимо сделать для того, чтобы любое ответственное лицо с базовым уровнем владения персональным компьютером могло с легкостью работать с данным приложением.

1 АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧИ

1.1 Постановка задач

Целью данного курсового проекта является разработка приложения для картинной галереи.

В ходе выполнения данного курсового проекта будет спроектировано программное обеспечение для картинной галереи. Картинная галерея должна обеспечивать работу и хранение необходимой информации.

Приложение должно содержать:

1) Главное меню.
2) Окно для работы с выставками.
3) Окно для работы с сотрудниками.
4) Окно для работы с отчётами.

Система должна быть представлена в двух частях:

1) База данных, основанная на системе управления базами данных (СУБД), которая поддерживает серверный режим работы.
2) Клиентское приложение.

1.2 Описание предметной области

Обязательно необходимо реализовать следующие действия для работы с данным приложением:

1) Работа со списком выставок.
2) Просмотр списка выставок и их подробной информации.
3) Работа с сотрудниками галереи.
4) Просмотр информации о сотруднике.
5) Формирование отчёта о выставке.

Работа с записями о сотрудниках и выставках предприятия предоставляет из себя такие возможности как:

1) Создание записи о работнике и выставке.
2) Удаление записи о работнике и выставке.
3) Редактирование записи о работнике и выставке.

Также система отдела кадров должна содержать сведения о следующих объектах:

1) Сотрудники.
2) Выставки.

Так же доступ к системе должен осуществляется после процедуры аутентификации.

1.3 Выбор и обоснование средств и методов решения задач

При моделировании картинной галереи будет использоваться унифицированный язык моделирования UML.

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web – приложений и даже встроенных систем реального времени.

UML – это язык для визуализации, специфицирования, конструирования и документирования артефактов программных систем. Язык моделирования, подобный UML, является стандартным средством для составления «чертежей» программного обеспечения. [2]

Так в процессе разработки моделей системы будет использовано CASE – средство EnterpriseArchitect8, которое представляет собой мощный набор UML инструментов для бизнеса и системного анализа, охватывающий все стадии разработки программного обеспечения: анализ, разработку, тестирование и поддержку.

Для реализации приложения будем использовать компилятор IDE Microsoft VS 2019.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Из языков программирования был выбран C#. У языка C# есть много преимуществ перед другими языками программирования, что позволяет решать с его помощью практически любые задачи.

C# – это объектно-ориентированный язык, который позволяет создавать модульные программы, исходный код которых может использоваться многократно.

Для управления данными информационной системы будет использоваться MySQL. MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой SunMicrosys-tems, которая ранее приобрела шведскую компанию MySQL AB. [3]

2 СОЗДАНИЕ ФУНКЦИОНАЛЬНОЙ МОДЕЛИ ПО

2.1 Разработка диаграммы вариантов использования

В начале проектирования системы строится модель в форме, так называемой диаграммы вариантов использования (usecasediagram), которая описывает функциональное назначение системы.

Диаграммы вариантов использования (usecasediagrams) представляют собой графическое представление взаимодействия пользователя и компьютерной системы. Каждый вариант использования охватывает некоторую очевидную для пользователей функцию системы и решает некоторую дискретную задачу пользователя. Список всех вариантов использования фактически определяет функциональные требования к системе с помощью, которых может быть сформулировано техническое задание, поэтому диаграмма вариантов использования является необходимым средством при анализе требований, выдвигаемых по отношению к проектируемой системе.

Диаграмма прецедентов не слишком детальна – например, не нужно ожидать, что она смоделирует порядок, в котором выполняются шаги. Вместо этого, правильная диаграмма вариантов использования изображает общий обзор взаимосвязей между сценариями использования, субъектами и системами.

Варианты использования представлены в виде помеченного овала. Фигуры из линий представляют актеров в процессе, а участие актера в системе изображается линией между актером и вариантом использования (ассоциацией). Между вариантами использования определены отношение обобщения и две разновидности отношения зависимости – включения и расширения. Отношение включения означает, что базовый элемент явно включает поведение другого элемента. Отношение расширения означает, что базовый элемент неявно включает поведение другого элемента; это отношение применяется для моделирования выбираемого поведения программы.

На диаграмме вариантов использования изображен один актер, который является администратором данной системы, другой обычный сотрудник. Для взаимодействия им нужно пройти процедуру авторизации. После авторизации им предоставляется одна из возможностей таких как: выполнить работу с подготовкой выставке, в которую входит редактирование, удаление или же добавление данных о выставке, а так же есть возможность такое же манипулирование отчётами. Администратор в свою очередь обладает особой привилегией добавить, редактировать, или же удалить сотрудника галереи.

Итоговая диаграмма вариантов использования для разрабатываемого программного обеспечения, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении Б.

2.2 Оценка трудозатрат на разработку ПО на основе диаграммы вариантов использования

2.2.1 Определение весовых показателей действующих лиц

Все действующие лица системы делятся на три типа: простые, средние и сложные. В данном случае действующие лица (пользователь, сотрудник галереи и администратор галереи) являются сложными, так как они представляют личность, пользующуюся графическим интерфейсом (GUI). Подсчитанное количество действующих лиц каждого типа умножается на соответствующий весовой коэффициент, затем вычисляется общий весовой показатель A. Весовой коэффициент сложного действующего лица равен 3.

Действующих лиц в данном случае два, следовательно, общий весовой показатель A равен 6. [4]

2.2.2 Определение весовых показателей вариантов использования

Все варианты использования делятся на три типа: простые, средние и сложные, в зависимости от количества транзакций в потоках событий (основных и альтернативных). Для простого типа варианта использования (3 или менее транзакций) весовой коэффициент принимается равным 5, для среднего (от 4 до 7 транзакций) – 10, для сложного (более 7 транзакций) – 15. Сложность вариантов использования представлена в приложении А в таблице 2.

Общий весовой показатель равен:

UUCP = 11*5 + 3*10+3*15= 170.

В результате получаем показатель UUCP:

UCP = UUCP + A = 170 + 6 = 176.

2.2.3 Определение технической сложности проекта

Техническая сложность проекта (TCF – technicalcomplexityfactor) вычисляется с учетом показателей технической сложности. Каждому показателю присваивается значение Ti в диапазоне от 0 до 5 (0 означает отсутствие значимости показателя для данного проекта, 5 – высокую значимость). Значение TCF вычисляется по следующей формуле:

$TCF = 0.6 + (0.01 * (\sum TiBeci))$

Настройка показателей технической сложности проекта, выполненная в CASE– средстве EnterpriseArchitect8, представлена на рисунке 2.1.

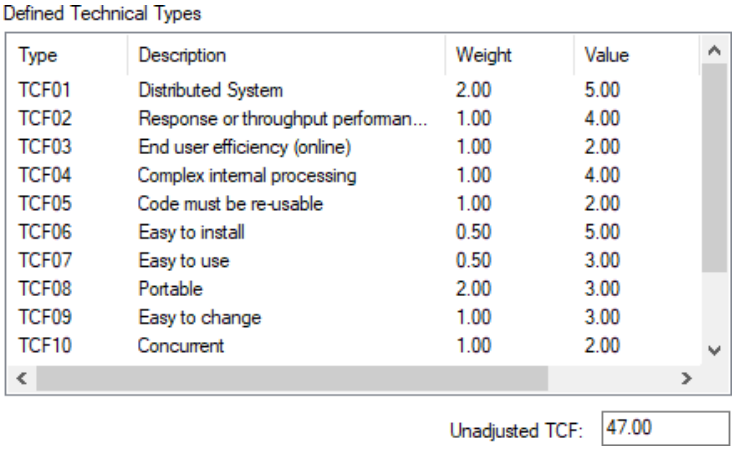

Рисунок 2.1 – Показатели технической сложности проекта

Вычислим значение TCF:
TCF = 0.6 + (0.01 * 47.00) = 1.07.

2.2.4 Определение уровня квалификации разработчиков

Уровень квалификации разработчиков (EF – environmentalfactor) вычисляется с учетом показателей F1 – F8 по формуле:

EF = 1.4 + (– 0.03 * (∑ FiBeci)).

Каждому показателю присваивается значение в диапазоне от 0 до 5. Для показателей F1 – F4 0 означает отсутствие, 3 – средний уровень, 5 – высокий уровень. Для показателя F5 0 означает отсутствие мотивации, 3 – средний уровень, 5 – высокий уровень мотивации. Для F6 0 означает высокую нестабильность требований, 3 – среднюю, 5 – стабильные требования. Для F7 0 означает отсутствие специалистов с частичной занятостью, 3 – средний уровень, 5 – все специалисты с частичной занятостью. Для показателя F8 0 означает простой язык программирования, 3 – среднюю сложность, 5 – высокую сложность. [5]

Настройка показателей уровня квалификации разработчиков, выполненная в CASE– средстве EnterpriseArchitect8, представлена на рисунке 2.2.

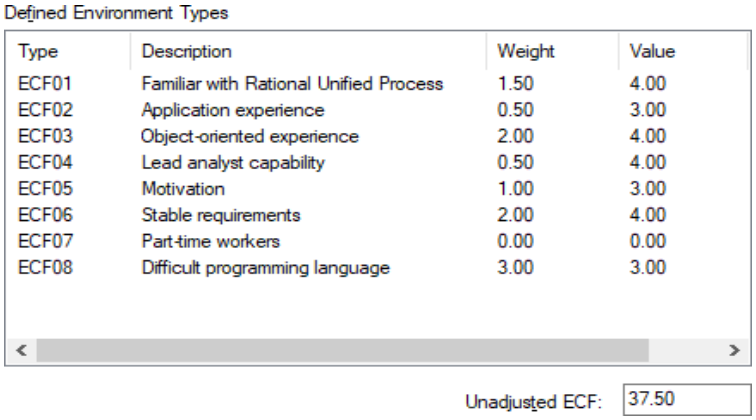

Рисунок 2.2 – Показатели уровня квалификации разработчиков

Вычислим значение EF:
EF = 1.4 + (– 0.03 *37.50) = 0.275.
В результате получаем окончательное значение UCP (usecasepoints):
UCP = UUCP * TCF * EF = 170* 1.07 * 0.275 = 50.

2.2.5 Оценка трудозатрат проекта

Для разрабатываемой системы примем 10 рабочих часов на одну UCP. Таким образом, общее количество рабочих часов на весь проект равно 10 * 50 = 500. Если принять стоимость часа работы равной 40 денежным единицам, то стоимость проекта будет равна 500*40 = 20000 денежным единицам.

2.3 Создание макета графического интерфейса пользователя

В процессе создания программного продукта одним из основных элементов, подлежащих разработке, является графический пользовательский интерфейс (GUI).

Графический интерфейс пользователя (GUI) – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

В отличие от интерфейса командной строки в GUI пользователь имеет произвольный доступ к видимым объектам с помощью устройства ввода. Зачастую элементы интерфейса реализованы в виде метафор и отображают их свойства и назначение для облегчения понимания пользователя.

Исходя из основных сценариев вариантов использования, создадим макеты экранных форм проектируемого программного обеспечения (приложение В), а именно: окно авторизации, главное меню для сотрудника галереи и администратора, окно для добавления/редактирования/удаления данных и сотрудников галереи в котором можно будет просматривать информацию о данных, сотрудниках и окно манипулирования отчётами.

Основной упор при проектировании интерфейса приложения был сделан на простоту и понятность для обычного ПК юзера.

3 СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ПО

3.1 Разработка диаграммы классов

Диаграммы классов при моделировании объектно-ориентированных систем встречаются чаще других. На таких диаграммах отображается множество классов, интерфейсов, кооперaций и отношений между ними. [6]

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Кроме того, диаграммы классов составляют основу еще двух диаграмм – компонентов и развертывания. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы.

3.1.1 Описание модели

В процессе проектирования программного обеспечения будем придерживаться трехуровневой модели архитектуры разрабатываемого приложения. Трех уровневое приложение включает следующие уровни:

Уровень данных, он содержит в себе смоделированные классы, являющиеся предметными сущностями проектируемой информационной системы.

Бизнес-уровень – второй уровень, работает как мост между уровнем данных и уровнем интерфейса. Бизнес-уровень представляет собой классы управления, с помощью которых будет реализовываться функционал системы.

Уровень интерфейса – уровень, на котором пользователи взаимодействуют с приложением. На диаграмме уровня интерфейсов находятся только классы, реализующие графический интерфейс проектируемого приложения. Каждый класс – экранная форма, спроектированная в рамках курсовой работы.

Диаграмма пакетов, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении Г (Рисунок Г.1).

3.1.2 Проектирование уровня данных

Уровень данных в нашем случае представляет собой собственно базу данных.

Исходя из краткого анализа предметной области, были выявлены такие сущности как:

1) «Worker».
2) «Position».

3) «Exhibition».
4) «Report».

Так «Worker» представляет собой класс, который содержит всю информацию о сотруднике галереи и предоставляет информацию о нем. «Position» класс содержащий информацию о должности сотрудника. «Exhibition» класс, который хранит информацию о выставке. В классе «Report» содержится информация об отчёте выставки.

В результате построена диаграмма классов уровня данных, выполненная в CASE – средстве EnterpriseArchitect8, которая представлена подробнее в приложении Г (Рисунок Г.2).

3.1.3 Проектирование уровня интерфейса и бизнес-логики

На диаграмме классов уровня интерфейса опишем классы экранных форм, эскизы которых были построены в пункте 2.3. К ним относятся такие классы, как класс «Authorization», «MainMenu», «WorkWithExhibition», «Report»,«Exhibitions», «Add/Edit Exhibition», «WorkersInfo», «Add/Edit Worker». Данные контролеры содержат обработчики событий для всех элементов управления. Так на этой диаграмме показаны классы, отвечающие за поток данных в системе и взаимодействие их между всеми компонентами системы.

Уровень бизнес-логики будет представлять собой набор классов-контроллеров для всех форм приложения. Так в рамках данного курсового проекта были созданы такие классы как:

1) «Admin».
2) «Worker».
3) «Interface».
4) «DB».

Диаграмма классов уровня интерфейса и бизнес-логики, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении Г (рисунок Г.3, рисунок Г.4).

3.2 Разработка диаграмм последовательностей

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. При этом учитываются два аспекта: во-первых, взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности. Во-вторых, можно рассматривать структурные особенности взаимодействия объектов.

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. На этих диаграммах изображаются только те объекты, которые непосредственно участвуют во

взаимодействии т.к. ключевым моментом является именно динамика взаимодействия объектов во времени и не используются возможные статические ассоциации с другими объектами. [7]

На диаграмме работы с системой показано как пользователь взаимодействует с системой. На этой диаграмме изображено, что пользователь может пройти авторизацию. После авторизации, в зависимости от авторизованного аккаунта, пользователю предлагается окно сотрудника, в котором происходит основные действия.

На диаграмме окна сотрудника показано, что сотрудник может выбрать одно из представленных действий и система его обработает.

Так более подробно с диаграммами последовательности вы можете ознакомиться:

Диаграмма последовательностей процесса работы с системой, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении Д (рисунок Д.1).

Диаграмма последовательностей окна сотрудника, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении Д (рисунок Д.2).

3.3 Разработка диаграмм деятельности

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельностей, а действий, и в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой, операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому. В контексте языка UML деятельность (activity) представляет собой некоторую совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к некоторому результату или действию (action). На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на

результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения. [8]

Так в рамках данного курсового проекта была разработана диаграмма деятельности, моделирующая процесс обработки данных всей информационной системы, и моделирующая процесс обработки информации при управлении системой.

Так на диаграмме деятельности моделирующей процесс обработки данных всей информационной системы показано, что пользователю при входе в систему предоставляется возможность произвести вход в систему. Если пользователь решит авторизоваться, то после успешного входа если его аккаунт принадлежит сотруднику галереи предоставляется возможность управления системой. Если же пользователь прошел авторизацию под аккаунтом администратора, он может получить актуальный список выставок, отчётов и информацию о сотрудниках.

На диаграмме обработки информации при управлении системой сотрудник выбирает одно из действий, после этого введенные данные обрабатываются и вносятся в БД.

Так более подробно ознакомиться с диаграммами последовательности вы можете:

Диаграмма деятельности моделирующей процесс обработки данных всей информационной системы, выполненная в CASE средстве EnterpriseArchitec8, представлена в приложении Е (рисунок Е.1).

Диаграмма деятельности процесса обработки выставки, выполненная в CASE средстве EnterpriseArchitect8, представлена в приложении Е (рисунок Е.2).

3.4 Разработка диаграмм состояний

Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Чаще всего диаграммы состояний используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы. Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдо состояния), которые изображаются соответствующими графическими символами. Дуги служат для обозначения переходов из состояния в состояние. Для понимания семантики конкретной диаграммы состояний необходимо представлять не только особенности поведения моделируемой сущности, но и знать общие сведения по теории автоматов. [9]

Так в рамках данного курсового проекта были разработаны диаграммы состояний, моделирующие процесс обработки информации во всей проектируемой информационной системе и обработки информации при аутентификации.

На диаграмме обработки информации во всей проектируемой информационной системе показано, что при запуске системы она предоставляет возможность пользователю авторизироваться под ролью сотрудника или же администратора. Если пользователь авторизовался под аккаунтом с правами сотрудника галереи, то система ждет выбора его действия. Сотрудник галереи или же администратор может выбрать удаление, редактирование или же добавление выставки в систему, а также такие же действия, но с отчётами, администратор может манипулировать сотрудниками галереи.

На диаграмме аутентификации показано, что система переходит в состояния ожидания ввода. После этого введенные данные проверяются на корректность и ищутся в БД. Если эти действия были выполнены успешно, система переходит в следующее состояние.

Более подробно с диаграммами состояний разрабатываемого программного обеспечения вы можете ознакомиться:

Диаграмма состояний процесса обработки информации во всей проектируемой информационной системе, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении Ж (рисунок Ж.1).

Диаграмма состояний обработки информации при аутентификации. Результат построения этой диаграммы в CASE – средстве EnterpriseArchitect8, представлен в приложении Ж (рисунок Ж.2).

4 СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ПО

Название «физическая модель» в терминологии языка UML отличается от общепринятой трактовки этого термина в общей классификации моделей систем. В последнем случае под физической моделью системы понимают некоторую материальную конструкцию, обладающую свойствами подобия с формой оригинала.

К достоинствам физических моделей относятся наглядность, надежность результатов наблюдений, возможность своевременно внести коррективы. К недостаткам – сложность их организации и осуществления, потребность в одновременном использовании значительного числа исследователей, а также сравнительно высокие затраты на создание уникальных моделей.

В языке UML для физического представления моделей систем используются так называемые диаграммы реализации, которые включают в себя две отдельные канонические диаграммы: диаграмму компонентов и диаграмму развертывания. [10]

4.1 Разработка диаграммы компонентов

Все рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Диаграмма компонентов описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов. [11]

Программа состоит из исполняемого файла приложения и базы данных. Для компиляции исполняемого файла требуются файлы исходного кода, для которых в свою очередь требуются библиотеки языка C#. Исходный файл имеет доступ к таким интерфейсам, которые в свою очередь связаны с БД.

Диаграмма компонентов разрабатываемого программного обеспечения, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении З (рисунок З.1).

4.2 Разработка диаграммы развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты – экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывании не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются.

Узел (node) представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом, а также другие механические или электронные устройства, такие как датчики, принтеры, модемы, цифровые камеры, сканеры и манипуляторы.

Графически на диаграмме развертывания узел изображается в форме трехмерного куба. Узел имеет собственное имя, которое указывается внутри этого графического символа. [12]

На диаграмме развертывания разрабатываемого программного обеспечения показано, что система состоит из двух узлов. Первым узлом является база данных, а второй это непосредственно устройство пользователя управляема операционной системой Windows. Так же на диаграмме показано, что между этими узлами существует связь посредством протокола TCP/IP.

Диаграмма развертывания разрабатываемого программного обеспечения, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении И (рисунок И.1)

5 ОПИСАНИЕ РЕАЛИЗАЦИИ И ТЕСТИРОВАНИЯ ПО

5.1 Детальная реализация функциональных частей ПО

Программная реализация проекта выполнена посредством языка программирования C#. Для реализации приложения были разработаны все необходимые окна.

При запуске информационной системы мы попадаем на страницу с авторизацией (рисунок 5.1).

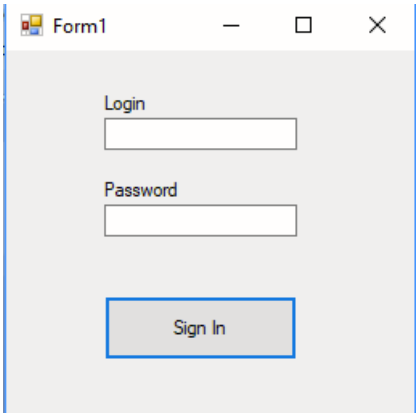

Рисунок 5.1 – Окно авторизации

Форма авторизации содержит элемент button, отвечающий за вход в приложение, а также 2 формы ввода для логина и пароля.

Листинг 5.1 – Обработчик кнопки

```
1: private void LoginButton_Click(object sender, EventArgs e)
2: {
3: string login = loginField.Text;
4:  string password = passwordField.Text;
5: DataTable userTable = new DataTable();
6: MySqlDataAdapter adapter = new MySqlDataAdapter();
7: string query = "SELECT * FROM `users` WHERE `login` = @login
AND `password` = @password ";
8:  MySqlCommand command = new
MySqlCommand(query,db.GetConnection());
9: command.Parameters.Add("@login", MySqlDbType.VarChar).Value =
login;
10: command.Parameters.Add("@password",
MySqlDbType.VarChar).Value = password;
11: adapter.SelectCommand = command;
12: adapter.Fill(userTable);
13: if (userTable.Rows.Count > 0)
```

```
14: {
15: MessageBox.Show(“Success!Authorization”);
16: this.Hide();
17: MainMenuForm menu = new MainMenuForm();
18: menu.Show();
19: }
20: else
21: MessageBox.Show(“No”);
22: }
```

После перехода, в главное меню у нас появляется возможность просмотреть или же провести действия с галерейной выставкой, вывод информации об выставке (рисунок 5.2).

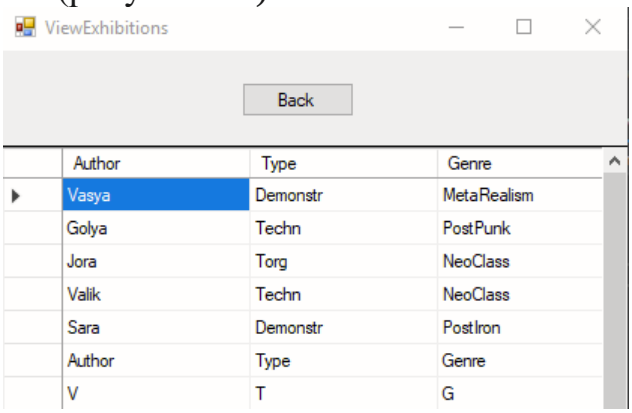

Рисунок 5.2 – Окно просмотра выставки

Так же мы можем выполнить различные действия с данной выставки от имени сотрудников данной галереи, меню редактирования выставкой (рисунок 5.3).

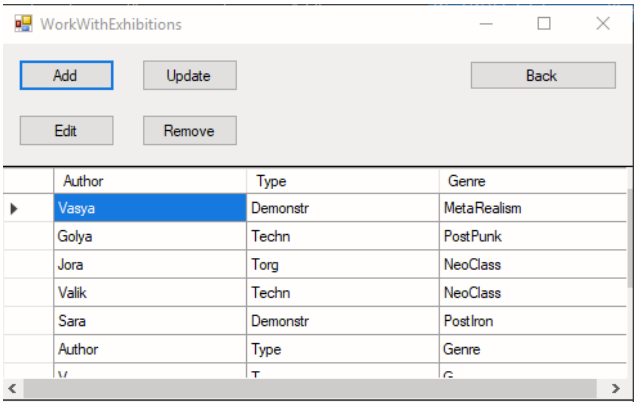

Рисунок 5.3 – Окно работы с выставками

Листинг 5.2 – Обработчик кнопки добавить

```
1: private void AddButton_Click(object sender, EventArgs e)
2: {
3: if(Author.Text == “”)
```

```
4: {
5: MessageBox.Show(“Enter “+Author.Name.ToString()+” Field”);
6: return;
7: }
8: if (Type.Text == “”)
9: {
10: MessageBox.Show(“Enter “ + Author.Name.ToString() + “
Field”);
11: return;
12: }
13: if (Genre.Text == “”)
14: {
15: MessageBox.Show(“Enter “ + Author.Name.ToString() + “
Field”);
16: return;
17: }
18: if (CheckUser())
19: return;
20: string query = “INSERT INTO `exhibitions`
(`author`,`type`,`genre`) VALUES (@author,@type,@genre)”;
21: MySqlCommand command = new MySqlCommand(query,
db.GetConnection());
22: command.Parameters.Add(“@author”, MySqlDbType.VarChar).Value
= Author.Text;
23: command.Parameters.Add(“@type”, MySqlDbType.VarChar).Value =
Type.Text;
24: command.Parameters.Add(“@genre”, MySqlDbType.VarChar).Value =
Genre.Text;
25: db.OpenConnect();
26: if (command.ExecuteNonQuery() == 1)
27: MessageBox.Show(“Was created”);
28: else
29: MessageBox.Show(“Wasnt created”);
30: db.CloseConnect();
31: this.Hide();
32: }
```

Данная программа имеет в своем распоряжении множество информационных сообщений, которые направлены на более удобное

использование. Также учтены возможные случаи некорректной работы программы, поэтому некоторое количество нештатных ситуаций сопровождается оповещениями.

5.2 Тестирование ПО

Для тестирования приложений используем SmokeTest (Дымовое тестирование). Данное тестирование представляет собой короткий цикл тестов, который проверяет, выполняет ли приложение основной функционал.

Результаты разработки и построения интерфейса форм протестированы и представлены в таблице 5.1.

Таблица 5.1 – Результаты тестирования программы

Тест	**Ожидаемый результат**	**Полученный результат**	**Результат**
Запуск программы	Запуск главного окна	Запуск главного окна	Успешно
Авторизация администратора	Вход в систему	Вход в систему	Успешно
Просмотр выставки	Вывод данных из БД	Вывод Данных из БД	Успешно
Кнопка перехода в редактирование выставок	Меню редактирования выставками	Меню редактирования выставками	Успешно
Добавление Выставки	Добавление данных о выставке	Данные добавлены	Успешно
Редактирование выставки	Редактирование данных о выставке	Данные редактированы	Успешно
Удаление выставки	Удаление данных о выставке	Данные удалены	Успешно
Авторизация без корректных данных	Неудачная авторизация	Вход в систему не выполнен	Успешно
Авторизация с корректными данными	Удачная авторизация	Вход в систему выполнен	Успешно

В результате проведения тестирования программного продукта не было выявлено ситуаций, при которых приложение совершало аварийную остановку или работало не корректно.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте были освоены технологии разработки программного продукта, а также спроектирована «Информационная система картинной галереи». Это приложение представляет собой программу для получения актуальной информации о выставках.

В ходе выполнения курсового проекта были созданы: функциональная, логическая, физическая модели ПО и следующие диаграммы:

– диаграммы вариантов использования;

– диаграммы классов;

– диаграммы последовательностей;

– диаграмма состояний;

– диаграммы деятельности;

– диаграммы компонентов;

– диаграммы развертывания.

Было использовано CASE – средство EnterpriseArchitect8, в котором содержаться все инструменты для проектирования, и отлично адаптировано под пользователя.

Разработанное приложение удовлетворяет всем требованиям, предназначенным для комфортной работы, и позволяет без проблем хранить и взаимодействовать с требуемыми данными.

Проведено полное тестирование приложения с разбором основных возможностей данного приложения.

В процессе выполнения курсового проекта были закреплены навыки по созданию различных диаграмм и макетов оконного интерфейса на языке UML 2.1, разработке приложений в Microsoft VS 2019, разработке целостной базы данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org. Дата обращения: 01.02.2020.
2. Википедия. Свободная энциклопедия. [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/GNU_Compiler_Collection – Дата доступа: 2.03.2020.
3. Официальный сайт справочной информации " Wikipedia ", [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/MySQL. Дата доступа – 28.04.2020 г.
4. Руководство по языку программирования C. – Режим доступа: https://metanit.com/c/tutorial/. Дата доступа: 16.03.2020
5. ClassServiceLoader <S> [Электронный ресурс] – Режим доступа: https://docs.oracle.com/c/6/docs/api/c/util/ServiceLoader.html. Дата обращения: 01.03.2020.
6. Википедия. Свободная энциклопедия. [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki /Диаграмма_состояний_(UML) – Дата доступа: 2.03.2020.
7. Наблюдатель[Электронный ресурс] – Режим доступа: https://refactoring.guru/ru/design– patterns/observer. Дата обращения 20.04.2020.
8. Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_классов. Дата доступа – 28.04.2020 г.
9. Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_последовательности. Дата доступа – 28.04.2020 г.
10. Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_деятельности. Дата доступа – 28.04.2020 г.
11. Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_состояний. Дата доступа – 28.04.2020 г.
12. Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_компонентов. Дата доступа – 28.04.2020 г.

ПРИЛОЖЕНИЯ