МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра программной инженерии**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**По теме:** «Разработка информационной системы

и прототипа виртуального музея на её основе»

Допущена к защите Выполнил:

Заведующий кафедрой: студент группы 381508

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Киселёв И.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись

Научный руководитель:

Доц. кафедры ПРИН, к. ф.-м. наук, Шапошников Дмитрий Евгеньевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород  
2019

Содержание:

**ВВЕДЕНИЕ**

В современном мире всё чаще говорят о вопросах связанных с глобализацией. Глобализация в широком смысле слова – это  процесс [всемирной экономической](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F), [политической](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F), [культурной](https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%BB%D1%8C%D1%82%D1%83%D1%80%D0%B0) и религиозной [интеграции](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F). С увеличением доступности и ростом числа пользователей сети интернет вопросы о всеобщей глобализации становятся всё более востребованными и всё более реализуемыми. По данным Internet World Stats на момент 9 мая 2019 года в сети интернет зарегистрировано 4,383,810 тысяч уникальных пользователей. Это порядка 56,8 % от общего числа населения Земли. Каждый второй житель Земли имеет возможность получить доступ к информации, источник которой находится за тысячи километров от него.

На протяжении всей истории искусство следовало за человечеством. Начиная от пещерных людей и вплоть до наших дней. Сохранение культурного наследия и обеспечения доступа к ней широких масс – одна из ключевых задач всеобщей глобализации. Использование цифровых технологий может улучшить производственный процесс в обычных музеях и художественных галереях, а так же значительно облегчить взаимодействие между человеком и культурным наследием всего человечества.

# Однако, чтобы экспонаты стали доступны для всех, необходимо тщательно заниматься их оцифровкой. Для того, чтобы оцифровать один экспонат экспертам может понадобиться много времени. По данным статьи Accessing Russian culture online: The scope of digitization in museums across Russia на момент 19 сентября 2018 года в Российской Федерации оцифровано лишь 13,86% от общего числа экспонатов находящихся в распоряжении музеев РФ. Наибольшее число экспонатов оцифровано в Центральной России (1.77 %), на Урале (3.2%) и Северо-Западном федеральном округе(1.32%). То есть Музеи России по большей части мало участвуют в процессе общей глобализации и цифровом сохранении культурного наследия.

# Помимо общей базы данных экспонатов, необходимо так же обеспечить и доступ к этой базе данных. По информации из того же источника средняя доля аналоговых коллекций воспроизводимых в цифровом виде по России всего лишь 2.15%. При том, что в среднем 18% музейных предметов имеют свои цифровые изображения для основной коллекции (без библиотечных книг и сопроводительной документации). То есть занести экспонаты в базу данных недостаточно. Необходимо так же обеспечить доступ к ним.

# В связи с вышеперечисленными факторами актуальность программного обеспечения подходящего не только для оцифровки культурно-значимых экспонатов, но и для предоставления пользователя доступа к ним не вызывает каких-либо вопросов. Для полноценного участия России в процессах культурной глобализации необходимо обеспечить доступные и легкие для усвоения инструменты для ускорения процессов оцифровки, так же позволяющие простым пользователям получить доступ к результатам этих процессов.

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

# Анализ предметной области и выявление текущих проблем.

# В декабре 2017 года документ под названием ReACH (Reproduction of Art and Cultural Heritage — «Воспроизведение искусства и культурного наследия») подписали ЮНЕСКО, Государственный Эрмитаж (Санкт-Петербург), Дворцовый музей (Пекин), Институт Варбурга (Лондон), Институт по сохранению культурного наследия Йельского университета (Нью-Хейвен), Лувр (Париж), Музей Ближнего Востока (Берлин), Смитсоновский институт (Вашингтон), V&A (Лондон) и художественный фонд Factum Arte (Мадрид). Этот международный документ призван, прежде всего, защитить интересы музеев и обеспечить сохранность культурного наследия для будущих поколений. Разработку документа поддержал советник Президента РФ по культуре Владимир Толстой и Министерство культуры РФ. В центре внимания ReACH — вопрос о создании цифровых копий работ при их документировании, особенно там, где произведения искусства оказались под угрозой разрушения или уничтожения.

# Правительством Российской Федерации был разработан специальный сайт ГОСКАТАЛОГ.РФ , на котором размещены все оцифрованные экспонаты. Любой желающий может получить доступ к любому из опубликованных экспонатов. Работа над общей базой началась ещё в 2016 году. По планам работа должна быть закончена к 2025 году. На момент 11 февраля 2019 в Госкаталоге находилось 9548291 экспонатов. При этом стоит помнить, что это всего лишь 2,15% от общего числа всех экспонатов находящихся в музеях РФ.

# Помимо огромного числа самих экспонатов, необходимо помнить, что качественная оцифровка экспоната может занимать огромное количество времени и ресурсов. Современная система учёта экспонатов не позволяет очень быстро заносить их в общую базу данных. Для регистрации одного экспоната необходимо потратить от получаса времени. Этот факт существенно замедляет рост числа оцифрованных экспонатов. В связи с этим одна из выявленных проблем – ускорение сохранения и регистрации общественно значимых объектов.

# Процесс оцифровки музеев и предоставление доступа к ним на западе продолжается более десяти лет с начала XXI века. Термин «цифровое курирование» впервые появился на научном семинаре, созванном Коалицией по сохранению цифрового пространства и Британским национальным космическим центром в Лондоне в 2001 году для обсуждения совершенствование открытой архивной информационной системы и обмен знаниями о цифровом курировании в различных областях. В конечном итоге стали появляться так называемые виртуальные музеи. Они являются своеобразной платформой, с помощью которой музеи общаются с клиентами пределами стен музеев. Многие известные музеи желают иметь свою виртуальную среду, либо создать собственные web-ресурсы со всей оцифрованной информацией. Подобные проекты позволят упростить составление экскурсий для гидов, а так же позволят посетителям музеев самостоятельно изучать историю каждого экспоната и составлять собственные маршруты движения по локациям музея с помощью портативных устройств. Эксперты полагают, что подобная возможность не отнимет потенциальных клиентов у самих музеев, а наоборот позволит привлечь больше посетителей. Цель такого ПО не заменять музей в современном его представлении, сколько дополнять его. Однако в Российской федерации существует серьёзная проблема с подобным программным продуктом. Наиболее известный и целостный аналог западному программному продукту принадлежит всемирно известному Санкт-Петербургскому Эрмитажу. Его интерфейс удобен и понятен см. *рисунок 1*. Сами точки на карте – место установки камер.

# 

Рисунок 1

# Каждая точка – это интерактивное место, из которого сделали снимок, используя специальную 360°-камеру. В конечном итоге экскурсия по такому музею аналогична экскурсии по городу с помощью Google Maps.

# Основная проблема заключается в том, что это – единичный случай. У большинства музеев России нет даже такой интерактивной карты. Небольшие музеи, как, например, музей А. М. Горького в Нижнем Новгороде, не обладают достаточными финансами и необходимой известностью, чтобы позволить себе что-то подобное. На лицо ещё две проблемы рассматриваемого вопроса всей российской музейной индустрии – отсутствие возможности узнать о музее больше у клиентов, или же дороговизна программного обеспечения, для увеличения осведомлённости у простого обывателя.

1. Существующие методы и инструменты.

# Одними из известных западных аналогов являются виртуальный музей Лувра на iPhone, а так же The Metropolitan Museum of Art для iPhone и Android. Так как это закрытые программные продукты, деталей их реализации не оглашается. Единственно, о чём можно точно сказать, оба они используют модель клиент-серверной архитектуры. Оба этих приложения на портативные устройства имеют маловесный клиентский интерфейс, который содержит в себе способ связи с удалённым сервером и грамотное преобразование и отображение информации хранящейся на этом удалённом сервере.

# Наиболее известный и полноценный аналог отечественного программного продукта – упомянутая выше виртуальная экскурсия по Эрмитажу «Виртуальный визит». В отличии от вышеописанных мобильных приложений этот программный продукт использует технологию web-сервисов. Основное отличие такого подхода – это отсутствие необходимости устанавливать что-то на своё устройство, а так же кроссплатформенность. Любое устройство имеющее свой браузер, может получить доступ и иметь возможность прогуляться по виртуальному Эрмитажу. Однако у них так же и много схожего. Любое web-приложение так же основано на клиент-серверной архитектуре. Только интерпритатором поступающей информации служит сам браузер. Выполняя программный код, web-браузер запрашивает некоторую информацию с сервера, а затем обрабатывает её и показывает пользователю изображение.

# Сервера могут быть как облачными, так и находиться локально. Основным плюсом облачных серверов является практически бесперебойная доступность к ним, а так же современные способы борьбы с DDos-атаками и постоянный backup сервиса. К сожалению, за такие удобства зачастую приходится так же платить аренду. Локальные сервера позволяют хранить определённое количество информации, ограниченное возможностями машины( группы машин ). К тому же локальные сервера из-за каких-либо технический проблем могут быть временно или насовсем выведены из строя. Несмотря на кажущиеся неудобства для них доступны все возможности, которые доступны и для облачных серверов, однако в таком случае разработчику необходимо задумываться об их самостоятельной реализации.

Подобные программные продукты зачастую используют клиент-серверный подход из-за ряда его преимуществ:

* Данная архитектура позволяет не хранить огромное количество информации на каждой локальной машине, чтобы успешно пользоваться возможностями приложения. Это позволяет значительно уменьшить размер самого приложения.
* Одним из плюсов является удобство обновления получаемой информации. Достаточно изменить информацию на сервере, чтобы все клиенты начали получать обновлённую информацию.
* Возможность менять и модернизировать клиент и серверную часть приложения обособленно друг от друга. Зачастую, стараясь угодить пользователю, разработчик может выпускать частые обновления клиентской части, при этом совершенно не изменяя серверную составляющую

Однако у такого подхода есть серьёзный минус – приложение окажется бесполезным, если у клиента не будет доступа к сети Интернет. Но зачастую плюсы значительно перевешивают этот минус.

1. Выбор методов и инструментов для решения выявленных проблем.

Разрабатываемый мной программный продукт позволит создать более дешёвый аналог описанных выше примеров. Как и в аналогах, наиболее удачная и правильная архитектура для этих целей – клиент-серверная архитектура. Помимо приведённых выше преимуществ, она позволит удалённо подключаться к базе данных каждому из клиентов будущей системы. Помимо этого преимущество клиент-серверного взаимодействия в том, что к одному и тому же серверу, а значит и к одной и той же базе данных, возможно будет подключиться с разных версий и разных операционных сред, что обеспечит хорошую кроссплатформенность системы в будущем.

За основу базы данных будет взята популярная PostgreSQL. Её главными преимуществами является её реляционность, её поддержка большого списка типов хранимых данных, а так же её удобство при развёртывании и подключении к ней. Так же PostgreSQL является очень гибкой. Она позволяет создавать новые типы данных, функции. Имеет достаточные максимальные размеры таблиц, до 32 Tb. Число строк в таблице не ограничено. При этом максимальный размер строки 1.6 Tb. Для сравнения, MySQL и MariaDB имеют ограничение размера строк всего в 65 535 байт.

Для передачи данных по сети будет использован протокол TCP. В отличии от протокола UDP, TSP обеспечивает гарантированную доставку всех отсылаемых пакетов в нужном порядке. То есть пропадает необходимость проверки данных на целостность. Но в отличии от UDP такой способ общения между клиентом и сервером занимает больше времени. Приложение должно обмениваться двумя типами информации: команды для взаимодействия с базой данных, а так же изображениями. Для первого типа данных скорость передачи не важна, так как размер передаваемых данных небольшой. Для передачи изображений крайне важно сохранения целостности. Эти два фактора и определили выбор протокола сетевого взаимодействия.

Для первой версии программы-прототипа подойдёт desktop-клиент. Для его написания был выбран язык C#, так как он наиболее удобен для создания подобных приложений на основе Windows Form. Основная задача прототипа – показать корректное взаимодействие и работу серверной части и обмен с базой данных. Именно поэтому язык и тип клиентской части не так важен, поэтому и был выбран наиболее простой в реализации и удобный desktop вариант.

1. Идеализированная модель будущей программы.

В первую очередь будущая программа должна обеспечивать быстрый доступ к ресурсам, хранящимся в базе данных. А так же обеспечивать быстрое их добавление изменение и любое другое взаимодействие с базой данных.

Чтобы решить проблему долгого размещения и большой затраты человеко-часов на размещение и регистрацию оцифрованных экспонатов на сайте Госкаталог.рф программа должна будет обеспечивать автоматизацию этого процесса.

Программа должна быть доступна на самых популярных операционных системах и платформах: Android, IOS, Windows, web-сервис. Это позволит увеличить число заинтересованных и осведомлённых посетителей.

Программа должна быть удобна в использовании не только в рамках одного виртуального музея, а должна позволить взаимодействовать музеям между собой ( например для передачи виртуальных экспонатов ).

Программный продукт должен позволять прокладывать собственные маршруты исследования музеев, а так же составлять рекомендуемые маршруты. При этом предусматривается реализация навигатора по выбранному пользователем музею.

Программа должна вести учёт пользователей, собирать статистику посетителей. Так же предусматривается несколько типов пользователей: посетитель, менеджер и администратор. Для каждого музея это должны быть разные люди. Менеджеры и администраторы смогут влиять на вид и содержание своего музея, а так же иметь доступ к общей статистике. Это позволит накопить и использовать в дальнейшем пользовательский опыт для улучшения самого музея и данного программного продукта.

Программа должна быть более доступной по цене и распространяться только для лицензированных музеев России.

Так же программа должна будет отображать не только сами экспонаты, но и их расположение в музее, как на 2D-карте, так и в 3D.

Помимо этого программный комплекс должен быть легко расширяемым, чтобы иметь возможность привносить в программу различные новшества и фичи.

**РАЗРАБОТКА ПРОТОТИПА ПРОГРАММНОГО ПРОДУКТА**

1. Разработка требований к прототипу
   1. Функциональные требования

Функциональные требования посетителя:

* Программный продукт должен отображать текущее состояние схем всех доступных этажей.
* Программа должна корректно отображать места нахождения экспонатов на схеме
* Программа должна открывать доступные экспонаты. Корректно отображать их название, описание и изображения.
* Посетитель должен иметь возможность открыть и посмотреть любой из доступных ему этажей.

Функциональные требования менеджера:

Помимо возможностей пользователя программа должна предоставлять менеджеру следующие возможности:

* Авторизация в системе, при наличии логина и пароля
* Смена пароля для текущей учётной записи
* Создание нового этажа
* Помещение неактивных этажей в список активных
* Удаление активного этажа из списка активных
* Редактирование этажа
* Открытие любого созданного ранее этажа
* Создание новой точки экспонатов на выбранном этаже
* Удаление точки экспонатов
* Создание экспоната
* Добавление к создаваемому экспонату изображения
* Добавление свободного экспоната к точке расположения экспонатов
* Снятие экспоната с точки расположения

Функциональные требования администратора:

Помимо возможностей менеджера программа должна предоставлять администратору следующие возможности:

* Удаление экспоната
* Удаление этажа
* Создание нового аккаунта менеджера
* Удаление аккаунта менеджера
  1. Системные требования
     1. Клиент:

Минимальные системные требования:

* Объём оперативной памяти 512Мб.
* Операционная система Windows OS.
* Свободное место на диске 100Мб.
* Двухъядерный процессор с частотой ядра 1600МГц.
* Доступ в интернет. Минимальная скорость 512Кб/с.
  + 1. Сервер:

Минимальные системные требования:

* Доступ к сети интернет 3Мб/с.
* Четырёхъядерный процессор с частотой ядра 1600МГц.
* Операционная система Window OS.
* 1 Гб оперативной памяти.
* Свободное место на диске 128Гб.
  1. Программные требования:
     1. Клиент:
* Установленная Visual Studio 2017.
* Поддержка Windows Form.
* Установленная .NET Framework 4.7.
  + 1. Сервер:
* Установленная Visual Studio 2017.
* Установленная .NET Framework 4.7.
* Установленная версия PostgreSQL 11.
* Установленная программа pgAdmin версии 4.

1. Разработка архитектуры прототипа

Во время производной практики была поставлена задача разработки полной и подробной архитектуры будущего прототипа и всего взаимодействия системы. Помимо этого необходимо так же разработать формат базы данных, и взаимодействие её таблиц. Поскольку было решено, что наиболее подходящая архитектура – клиент-серверная, то и основой всего приложения будет именно клиент-серверная часть.

В серверной части должно располагаться средство принятия и передачи сообщений, а так же средства общения с базой данных, которая так же будет располагаться на сервере.

В клиентской части так же должно присутствовать средство передачи и принятия данных, для корректного общения с сервером. Помимо этого в клиентской части необходим обширный и детально продуманный графический интерфейс. В итоговом варианте вышла архитектура, которая показана на *рисунке 2*.

Для обеспечения верного взаимодействия и недопущения двойной интерпретации команд при общении клиентской и серверной части был введён список доступных команд, на рисунке он обозначается по краям схемы.

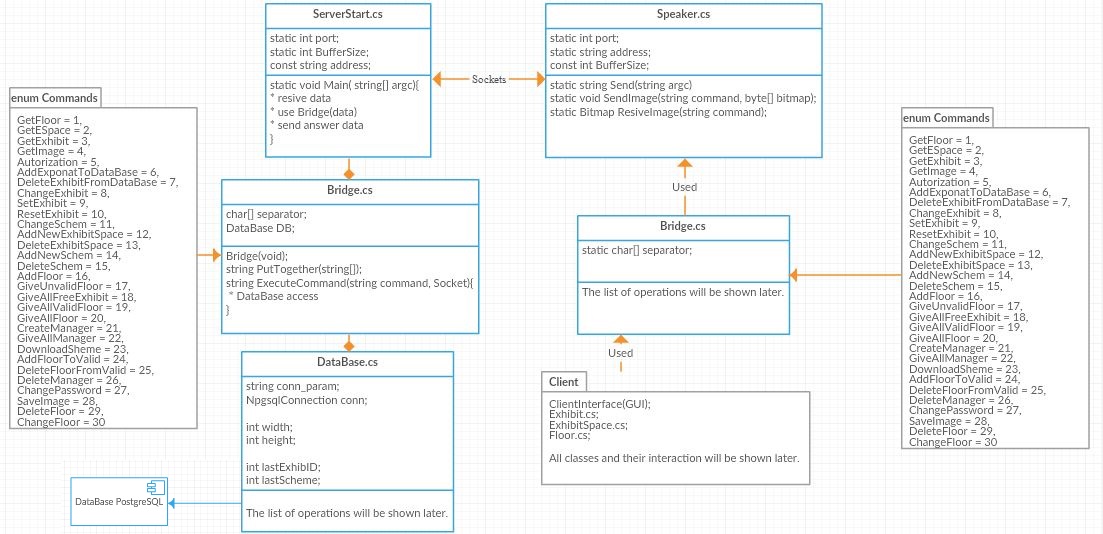


Рисунок 2

Так же по рисунку можно заметить, что между частями как сервера, так и клиента располагаются классы-прослойки. Для реализации будет использован паттерн проектирования мост, он же Bridge. Данный паттерн позволяет избежать перестроения всего кода, при изменении какой-либо одной части. В текущем проекте он крайне удобен, поскольку в дальнейшем это может обеспечить более удобный переход к кроссплатформенности приложения. Bridge, в клиентской части, предоставляет интерфейс взаимодействия графического клиента со средствами передачи информации. Его основной задачей является преобразование запросов GUI в запросы, которые возможно будет интерпретировать на сервере единственным образом. Соответственно Bridge на сервере интерпретирует полученные команды и формирует ответ, который потом Bridge-клиент передаст графическому интерфейсу в необходимом формате.

На *рисунке 2* показана общая архитектура и её взаимодействие, но не представлены полные интерфейсы и функционал некоторых классов. Это необходимо было сделать, чтобы избежать излишнего нагромождения. В ходе дальнейшего описания практики будут описываться составные части системы по отдельности для большей наглядности.

На *рисунке 3* показана схема, устройство и связи в базе данных PostgreSQL. База данных состоит из 7 таблиц, 6 из которых взаимосвязаны.

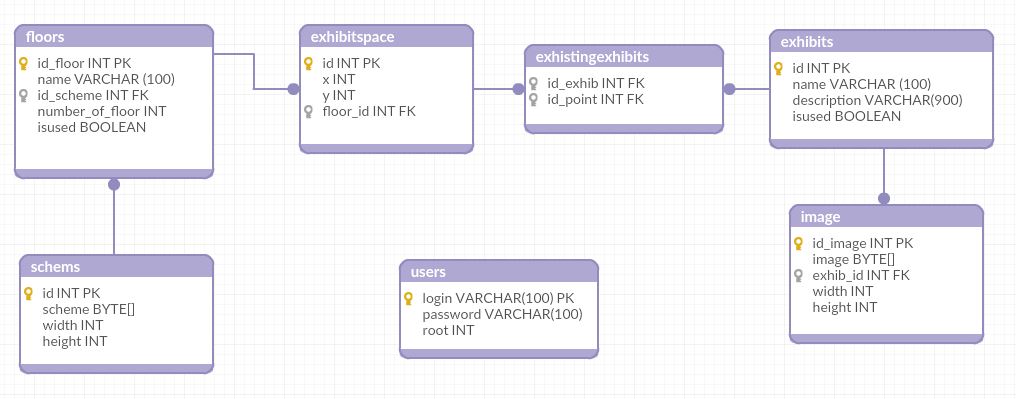


Рисунок 3

* Таблица пользователей (users) хранит в себе логины и пароли менеджеров и администраторов данного приложения, а так же их права доступа.
* Таблица этажей (floors) хранит в себе названия этажей, а так же схемы, которые соответствуют этим этажам. При этом если этаж является активным, то ему присуждается уникальный номер этажа.
* Таблица схем (schems) хранит в себе текущее графическое представление этажа, а так же размер этой схемы. Это позволит облегчить передачу и преобразование схемы. Эти схемы будут отображаться в клиентском интерфейсе.
* Таблица точек экспонатов (exhibitspace) хранит в себе координаты этой точки на схеме и номер этажа, которому она принадлежит.
* Таблица записей экспонатов (exhibits) хранит в себе информацию об экспонатах. Она содержит их имя, описание, а так же отображает используется ли экспонат на данный момент.
* Таблица активных экспонатов (exhistingexhibits) содержит в себе пары номеров точек экспонатов и номеров экспонатов. Присутствие экспоната в этой таблице означает, что данный экспонат на текущий момент можно увидеть, если выбрать соответствующую точку.

Таблица изображений (image) хранит в себе информацию об изображениях и сами изображения. Помимо размерности самого изображения в данной таблице так же хранится и информация о том, какому именно экспонату принадлежит это изображение.

Для взаимодействия с базой данных так же необходим класс, который будет формировать запросы к базе данных в зависимости от требования клиентов. Именно эту роль и выполняет класс DataBase. *Рисунок 4*

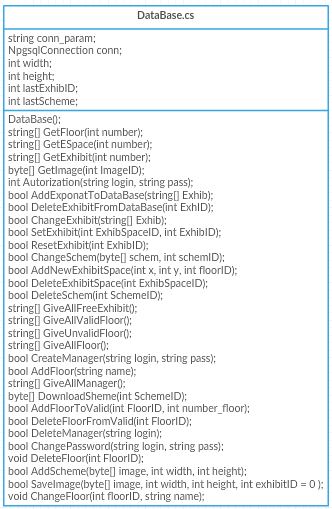


Рисунок 4

Каждая из представленных выше функций формирует уникальный запрос к базе данных. Это позволяет проводить разнообразные операции с базой данных: Чтение, запись, изменение и удаление чтобы обеспечить корректное взаимодействие базы данных с клиентскими запросами. На это с серверной частью можно закончить и перейти к клиентской части.

В отличии от серверного варианта, клиентский паттерн-мост имеет куда больше разнообразных функций, поскольку предполагается огромное число возможностей у клиента. Для каждой такой возможности создана специальная функция, которая формирует запрос так, чтобы его возможно было разобрать на сервере единственным способом. Клиентский Bridge показан на *рисунке 5*.

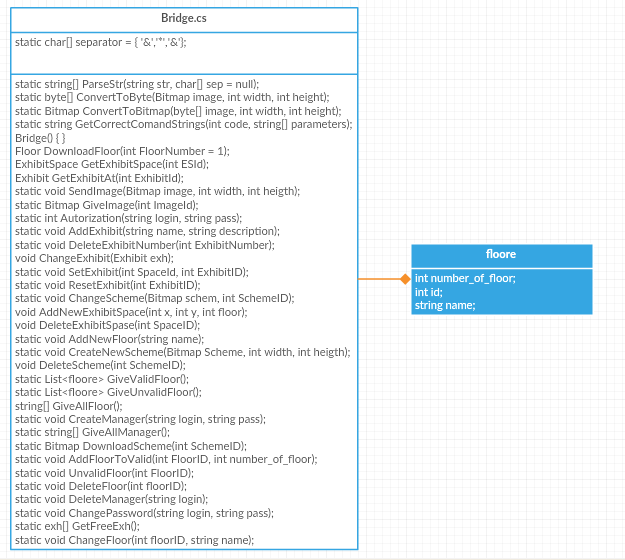


Рисунок 5

Можно заметить, что некоторые названия методов похожи на названия методов в DataBase.cs и это не случайно. Все спроектировано таким образом, чтобы методы Bridge обращались к методам DataBase через посредников и по сети, а так же было легче понимать саму структуру программы. Можно заметить, что Bridge использует некую структуру floore – эта структура является упрощенным представлением этажа, без необходимости подкачки схемы этого этажа. Это наиболее удобно при формировании списков активных и неактивных этажей.

Осталось лишь спроектировать последний элемент схемы, представленной на *рисунке 2* – саму клиентскую часть. Это наиболее трудоёмкая в плане понимания и реализации часть проектирования.

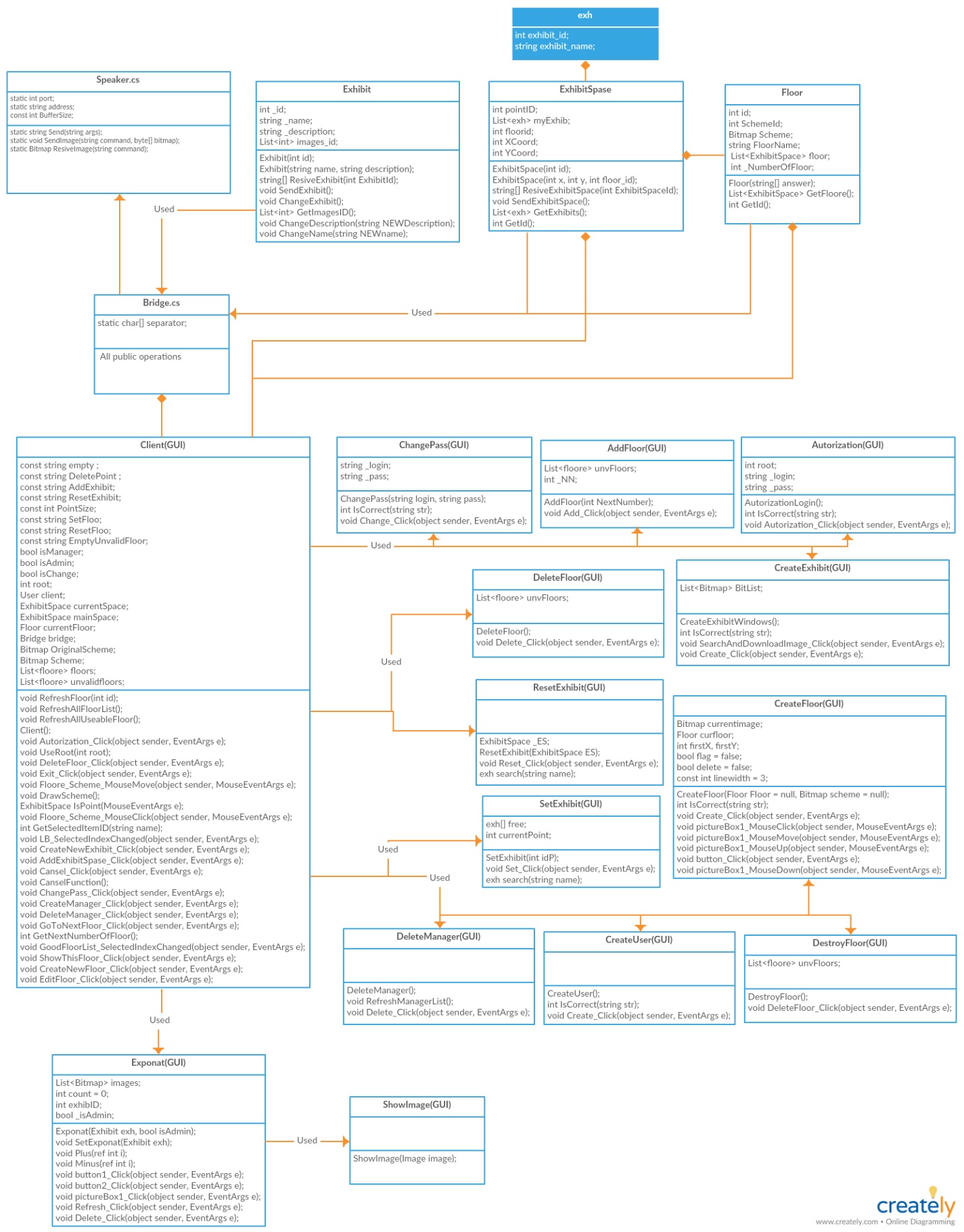
Результат можно увидеть на *рисунке 6*.

Рисунок 6

Основной класс во всей этой структуре – Client. Он представляет собой реализацию основного окна клиентского графического интерфейса. Помимо этого из него вызываются другие вспомогательные окна: Создание и изменение этажей, создание экспонатов, управление менеджерами будущего виртуального музея, открытие конкретных экспонатов, взаимодействие с точками экспонатов, авторизация и другие. Помимо графического представления клиентская часть так же содержит и вспомогательные классы:

* Exhibit – класс предназначенный для отображения экспонатов. Объект класса – это полноценный экспонат, который образуется из пришедшей с сервера информации. Класс содержит в себе имя экспоната, его описание, его идентификатор, а так же список принадлежавших ему изображений. Чтобы запросить конкретный экспонат, достаточно создать объект класса, указав в конструкторе идентификатор этого экспоната
* ExhibitSpace – класс предназначенный для корректного изображения точек расположения экспонатов на схеме этажей. Помимо этого функционала он так же служит контейнером для объектов типа exh. Exh это упрощённое представление класса Exhibit. Его поля содержат только имя и идентификационный номер соответствующего имени экспоната. Таким образом сокращается количество памяти, необходимого для работы программы, а так же ускоряется работа всего приложения.
* Floor – класс-основа представления графического интерфейса. Он хранит в себе название этажа, схему этого этажа, а так же все точки местоположения экспонатов на этом этаже.

За счёт этих трёх классов и формируется корректная работа приложения и визуализация текущего виртуального музея.

Некоторые окна, как вспомогательные, так и основное не могут быть использованы без этих классов. Так, окно Exponat является графическим представлением класса Exhibit, а принадлежащие Exhibit изображения мелко отображаются в Exponat, а более крупно и детально они отображаются в окне ShowImage.

Таким образом была разработана подробная архитектура будущего прототипа виртуального музея, на основе базы данных и технических требований к прототипу. Использование паттерна проектирования Мост позволит в дальнейшем проводить расширение функционала данного прототипа без ущерба всей системе в целом, а так же позволит обеспечить кроссплатформенность в будущем.

1. Разработка программного кода

Во время проектирования основным вопросом, помимо самой архитектуры, так же был выбор языка программирования для системы. Выбор был между двумя вариантами: C# и Java.

Язык программирования Java имел ряд преимуществ перед C#, ключевые из них – простой и понятный интерфейс сетевого взаимодействия, легкореализуемый и более привычный на тот момент параллелизм в серверной части.

C# в свою очередь, несколько уступая в простоте этих возможностей, обеспечивал более удобный интерфейс создания графического клиентского приложения.

Так как взаимодействие с базой данных у обоих языков реализовано похоже, удобство разработки desktop-приложения на C# стало основным критерием выбора этого языка.

Помимо языка программирования так же возникла необходимость использовать систему контроля версий GIT для более удобной и безопасной разработки.

* 1. Начало разработки

На начальном этапе разработки основной задачей была реализация структуры, показанной на *рисунке 2.* Первым делом необходимо было обеспечить сетевое взаимодействие клиентской и серверной части на локальном хостинге. Когда появилась возможность передавать по сети строки, началась реализация конкретных методов и, в частности, паттерна-проектирования Мост для обеих сторон клиент-серверного приложения.

Отдельным этапом можно назвать установку и настройку базы данных и ключевых таблиц, а так же обеспечение взаимодействия между базой данных и серверной частью.

Первый вариант графического интерфейса был создан с одной лишь целью – проверить корректность вызова нужных функций в зависимости от запросов пользователя. Благодаря Bridge этот процесс не занял много времени. После прохождения этих этапов можно было переходить к более детальной разработки.

* 1. Формирование прототипа и приведение его к итоговому виду.

Первое, что решено было сделать – это создать второй вариант графического интерфейса. Так как текущая разработка – прототип, необходимости в особом дизайне, как таковой, не было. Поэтому всё было реализовано с использованием стандартных Windows Forms. Параллельно с разработкой итоговой версии GUI появилась необходимость минимального взаимодействия с базой данных – авторизация. Был создан первый аккаунт администратора, и в этот же день получилось авторизоваться в системе. После этого шага началась разработка трёх секций клиентского приложения – пользовательский, менеджерский и администраторский. Началась активная реализация структуры показанной на *рисунке 6*.

Параллельно с разработкой GUI велась разработка и взаимодействия с базой данных. Реализация возможности для клиента сопровождалась соответствующей реализацией и для базы данных. Таким образом ставились цели и была возможность сфокусироваться на конкретных функциональные особенностях.

Таким образом, постепенно архитектура становилась всё более обширной и всё больше стала походить на план.

* 1. Окончание разработки

Последним этапом реализации была реализация передачи и обработки изображения. В отличии от всех действий до этого реализация изображений значительно усложнялась тем, что требовала больших затрат памяти. Для сравнения, объём передаваемых данных по сети для обычной команды была максимум 2 Кб, то при передаче изображения необходимо было минимум 250 Кб. Помимо этого изображение на начальных этапах передавалось в некорректном виде. Это приводило к тому, что восстановить это изображение при получении с базы данных было очень трудно.

Специально для передачи графических файлов были реализованы несколько дополнительных функций. Преобразование изображения в массив байт, преобразование массива байт в изображение, а так же передача массива байт по сети на сервер и получение подобного массива байт. Таким образом, передача изображения на локальном хосте была закончена и графический интерфейс обрёл свой финальный вид.

Но на этом разработка так же не закончилась. Для дальнейшего тестирования приложения необходимо было наладить связь с сервером используя интернет. При интернет соединении использовался внешний ip-адрес, который принадлежал wi-fi роутеру. Внутри локальной wi-fi сети было одновременно несколько устройств, поэтому пришлось использовать переадресацию вызова. Финальная цепочка связей выглядела так, как показано на *рисунке 7*:

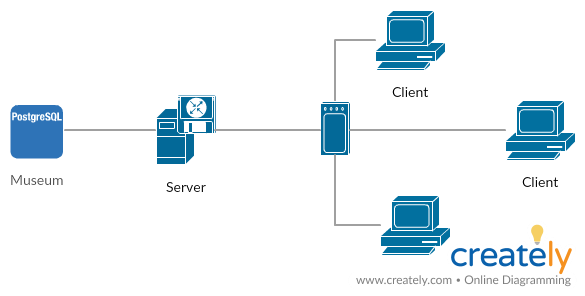


Рисунок 7

Устройство по центру – домашний wi-fi. Для полноценной системы такой вариант не подойдёт. Намного лучше будет передача данных серверу напрямую. Однако для прототипа данная схема подходит.

Таким образом разработка прототипа была закончена. Осталось его отладить и протестировать.

* 1. Отладка

Основной задачей отладки было найти как можно больше багов и исправить найденные проблемы до того, как приложение начнут тестировать потенциальные пользователи. Имея в распоряжении два персональных компьютера, началось тестирование функционала приложения, сначала в локальной сети, потом и через интернет соединение. Во время отладки были созданы несколько этажей, часть из них были отредактированы. Были созданы несколько экспонатов, часть из них с изображениями, часть без изображений. Созданы точки расположения экспонатов, на точки прикреплялись созданные экспонаты. Создан был менеджерский аккаунт. Созданные этажи назначались активными или неактивными. В итоге был проверен весь функционал описанный в пункте 1.1. Были выявлены и исправлены баги, связанные как с базой данных, так и с графической частью. Так же во взаимодействие с базой данных были внесены некоторые изменения, теперь удаление экспонатов влияло на все связанные таблицы. Таким образом при удалении чего-то из одной таблицы, при необходимости, удалялись связанные с этой записью пункты в других таблицах.

После завершения отладки можно было говорить об окончании работы над прототипом. Примерный объём кода прототипа: 3200 строк.

На основе готового приложения был разработан первый пример и на его основе было создано руководство пользователя и руководство менеджера, для более эффективной адаптации тестировщиков к приложению.

**ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ**

Были созданы 4 менеджерских аккаунта, два из которых были тестовыми. Менеджерам и администратору позволяется изменять схемы всех этажей, добавлять новые местоположения экспонатов, а так же управлять экспонатами, которые размещаются на этих точках. Помимо этого менеджеры и администратор имеют возможность создавать экспонаты, давать им названия, описание и загружать изображения. После создания их можно будет разместить на любой из местоположений экспонатов на активном этаже и они станут доступны для простых посетителей этого виртуального музея.

Для тестирования приложения решено было ограничится только ролями менеджера и пользователя. Роль администратора решено было оставить за создателем. Для каждой из ролей было написано руководство пользователя, которое объясняло ключевые принципы работы с программой.

**Графическое руководство пользователя**:

На *рисунке 8* представлена основная страница приложения. Выделенными областями обозначены:

1. Название выбранного этажа
2. Точка расположения экспонатов( в одной точке могут находится несколько экспонатов )
3. Схема или план этажа
4. Кнопка для обновления текущего этажа
5. Панель выбора следующего этажа. Пользователь может выбрать любой этаж из выпадающего списка, после чего нажать кнопку перейти. Приложение откроет выбранный пользователем этаж.
6. Кнопка авторизации. Если у вас есть логин и пароль, вы можете авторизоваться, нажав эту кнопку.

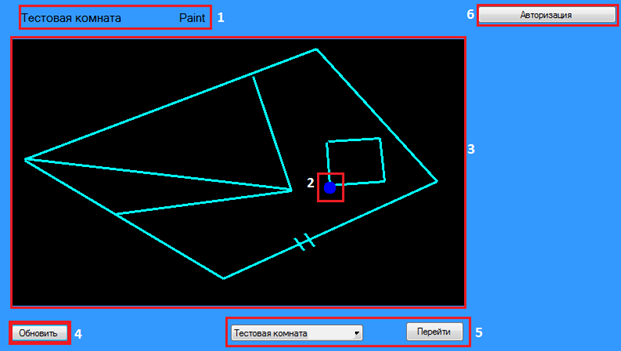


Рисунок 8

При наведении на одну из точек экспоната она изменит свой цвет, как показано на *Рисунке 9*.



Рисунок 9

При нажатии на подсвеченную точку пользователь увидит список экспонатов, которые находятся у этой точки. Пользователь может выбрать любой экспонат из предложенного списка. *Рисунок 10.*

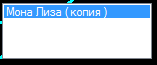


Рисунок 10

После нажатия на экспонат будет открыто окно этого экспоната. Пример такого окна можно увидеть на *Рисунке 11.*

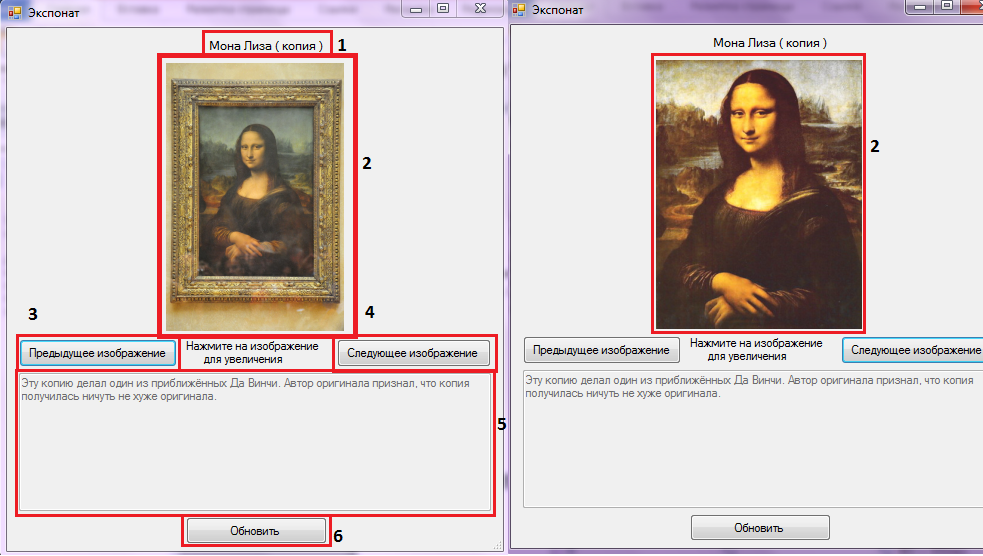


Рисунок 11

Цифрами на *рисунке 11* обозначены:

1. Название экспоната
2. Изображения и фотографии экспоната

3 и 4. Кнопки переключения между изображениями

5. Описание экспоната

6. Возможность обновить экспонат в случае, если что-то отобразилось некорректно.

При нажатии на изображение пользователь сможет более детально рассмотреть это изображение. При закрытии окна экспоната пользователь будет возвращён на стартовое окно.

Руководство менеджера занимает 7 страниц, при этом оно частично перекликается с руководством пользователя, поэтому в отчёте я отображу лишь ключевые моменты.

**Графическое руководство менеджера:**

Для авторизации необходимо нажать кнопку Авторизация и во всплывающем окне *рисунок 12* ввести свой логин и пароль (их сообщит вам администратор)

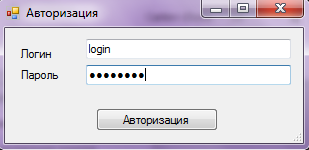


Рисунок 12

После успешной авторизации откроется основная страница приложения. На ней появится панель менеджера. *Рисунок 13*. Так же при успешной авторизации станут доступны новые возможности, о которых будет сказано позднее.

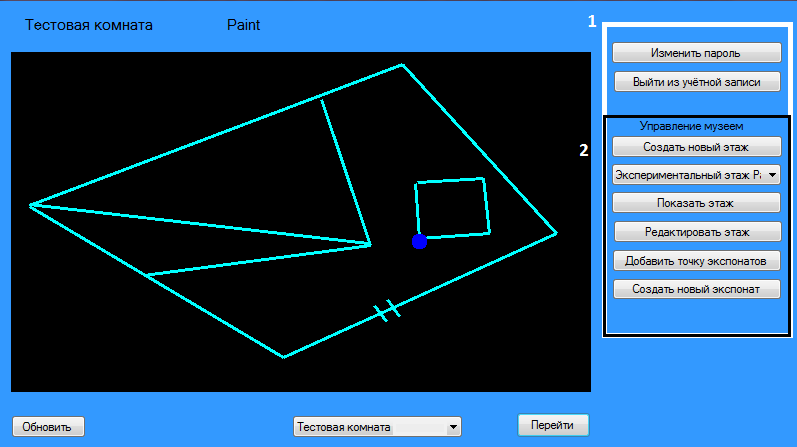


Рисунок 13

1. На панели менеджера есть 2 основные части:
   1. Панель взаимодействия со своим аккаунтом:

* Возможность изменить пароль своей учётной записи
* Выход из своей учётной записи
  1. Панель взаимодействия и музеем:
* Создать новый этаж - это возможность создать новую схему этажа, чтобы в дальнейшем ввести его в эксплуатацию. Окно создания показано на *рисунке 14.*

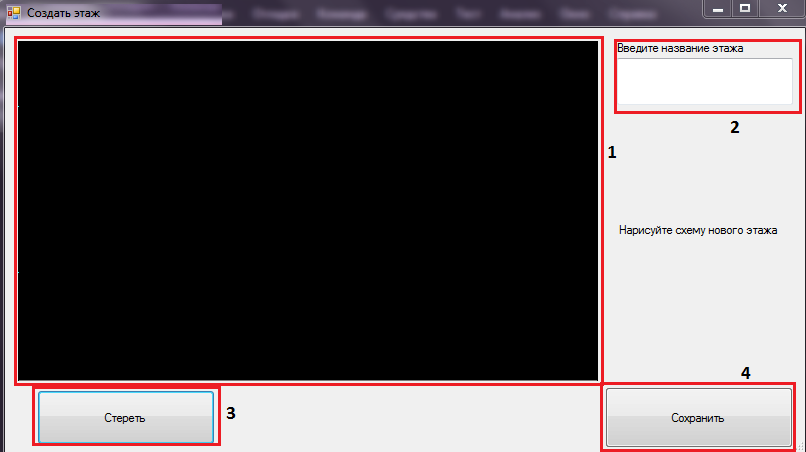


Рисунок 14

На *рисунке 14* цифрами обозначены:

1. Поле для редактирования. В этом поле используя прямые линии, возможно создать схему (план) этажа любой формы.
2. Название этого этажа. Менеджер может отредактировать название этажа.
3. Кнопка выбора текущего инструмента. У неё есть 2 состояния:

- Стереть (при нажатии активируется возможность стереть те линии, которые больше не требуются)

- При повторном нажатии на кнопку менеджер снова сможет менять схему, используя линии.

1. Кнопка позволяющая сохранить все изменения.

Важная часть руководства – это взаимодействие с другими частями интерфейса:

1. Взаимодействие с точками расположения экспонатов

При наведении и активации точки экспоната менеджер (в отличие от неавторизованного пользователя) сможет использовать дополнительные возможности: Удалить выбранную точку, повесить на это место ещё один экспонат, снять один из экспонатов. *Рисунок 15.*

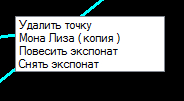


Рисунок 15

Если менеджер выберет повесить экспонат, ему будет показан список всех экспонатов, находящихся на складе. При выборе нужного экспоната и нажатии на кнопку выбранный экспонат станет доступным для всех пользователей, при открытии этой точки. При выборе строки “Удалить точку ” текущая точка будет удалена. Все экспонаты, принадлежавшие ей, будут автоматически перенесены на склад.

1. Взаимодействие со списком доступных этажей.

При открытие списка с доступными этажами у менеджера есть возможность изменить их состав, *рисунок 16*:

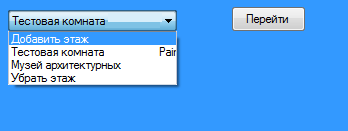


Рисунок 16

В отличии от неавторизованного пользователя менеджер может добавить или убрать этаж из списка активных. При выборе “добавить этаж” менеджеру предоставляется список всех неактивных этажей.

**Результаты удалённого тестирования**

Сервер, находящийся на моей локальной машине начал работу около 17:30. К тестированию я привлёк 10 заинтересованных людей, которые подключались из 2 городов одновременно. Это Нижний Новгород – 9 человек и Иваново – 1 человек. Сам сервер на момент тестирования находился в г. Ковров ( Владимирская область ) вместе с его создателем.

Всего сервер проработал с тестировщиками около трёх часов, два из которых он работал в стационарном режиме, постоянно обрабатывая запросы используя стабильное интернет соединение. Пиковая нагрузка на локальный сервер во время тестирования – 5 человек. По результатам тестирования была собрана статистика и отзывы тестировщиков. Им было задано несколько вопросов, результаты которых вы можете наблюдать на *рисунках 17-22*:



Рисунок 17



Рисунок 18



Рисунок 19



Рисунок 20

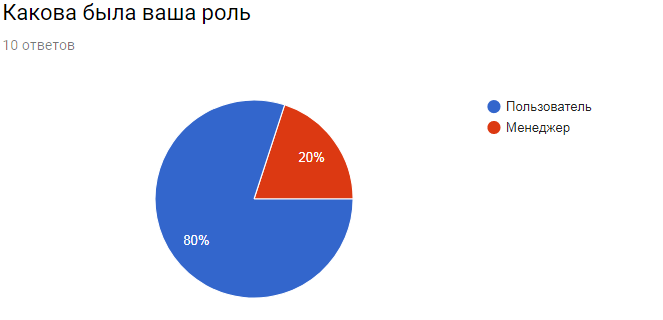


Рисунок 21

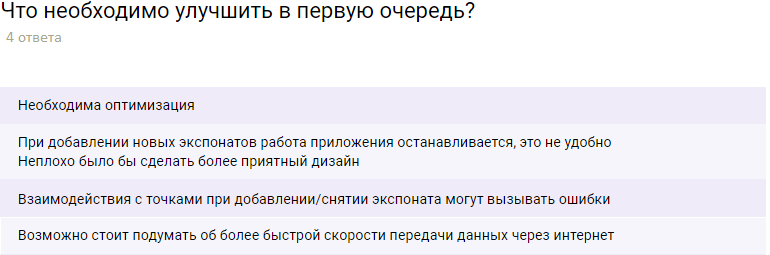


Рисунок 22

**Итоги тестирования**

По результатам тестирования можно сделать несколько выводов:

* Обмен данными через интернет работает
* Приложению необходимо оптимизировать передачу больших данных(изображений) клиенту. Частично на скорость передачи и принятия влияет скорость интернета, а так же сам способ подключения к интернету( 7 из 10 человек подключались через wi-fi роутер) Помимо этого стоит учитывать, что и сам сервер подключался к интернету через wi-fi.
* Выявилась ошибка связанная с графическим интерфейсом(один из отзывов тестировщика)
* Необходимо улучшать дизайн клиентской части.

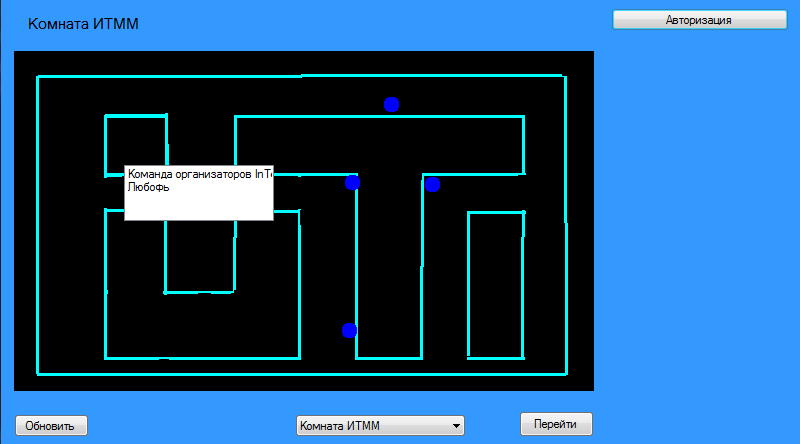
По результатам тестирования была создана комната студенческого совета Института Информационных Технологий Математики и Механики, в которой хранится около 20 фотографий. Эти фотографии размещены на разных точках и распределены по тематике и времени. Результат совместной работы можно наблюдать на *рисунке 23*. . 

Рисунок 23

Результаты тестирования выявили ключевые моменты, которые необходимо исправить прежде чем продолжать дальнейшее расширение функциональности приложения. Стоит помнить, что данный прототип разработан в первую очередь для проверки работоспособности серверной части и подключенной к ней базы данных. Учитывая данный факт можно говорить о том, что тестирование прошло успешно.