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**

Направление подготовки: «Программная инженерия»

Профиль подготовки: «Проектно-технологическая работа»

**ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

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

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

Рецензент:

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

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

подпись

Нижний Новгород  
2019

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 1.*

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

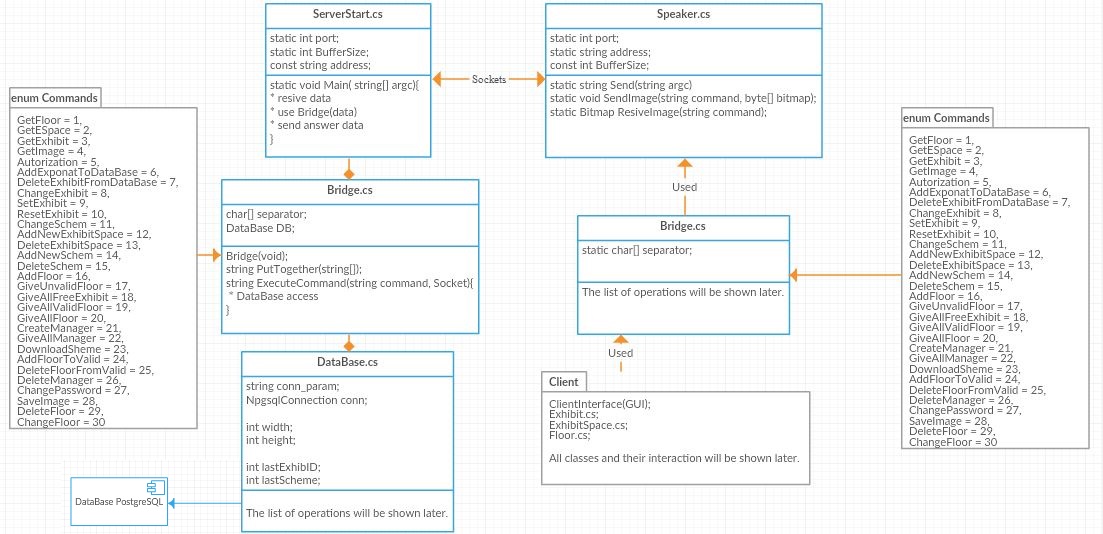


Рисунок 1

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 мультиплатформенности 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 1* 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 2* показана схема, устройство и связи в базе данных PostgreSQL. База данных состоит из 7 таблиц, 6 из которых взаимосвязаны.

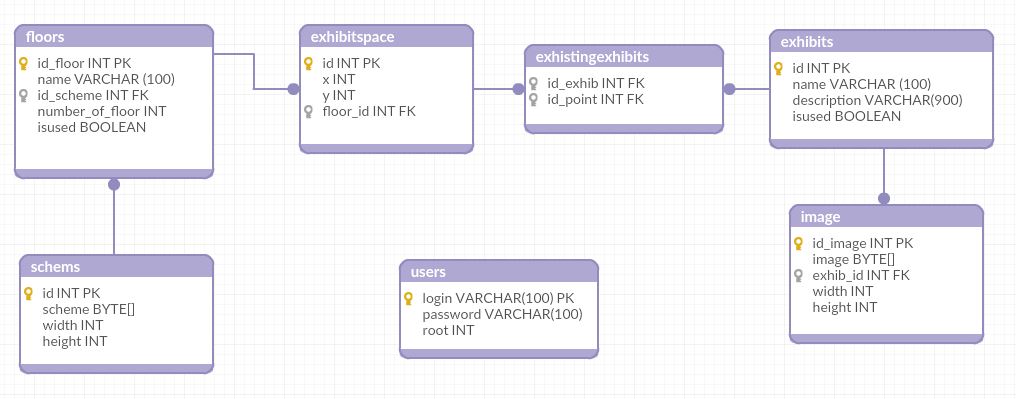


Рисунок 2

* 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 3*

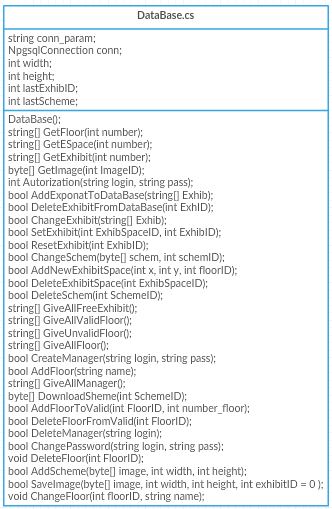


Рисунок 3

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 4*.

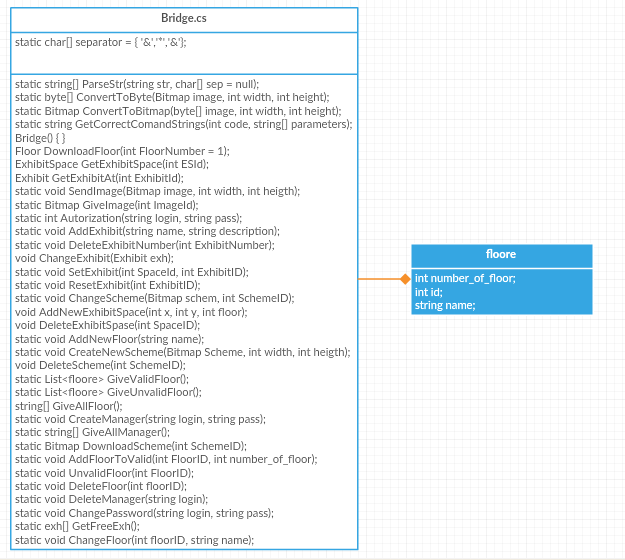


Рисунок 4

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 1* – саму клиентскую часть. Это наиболее трудоёмкая в плане понимания и реализации часть проектирования. Результат можно увидеть на *рисунке 5*.

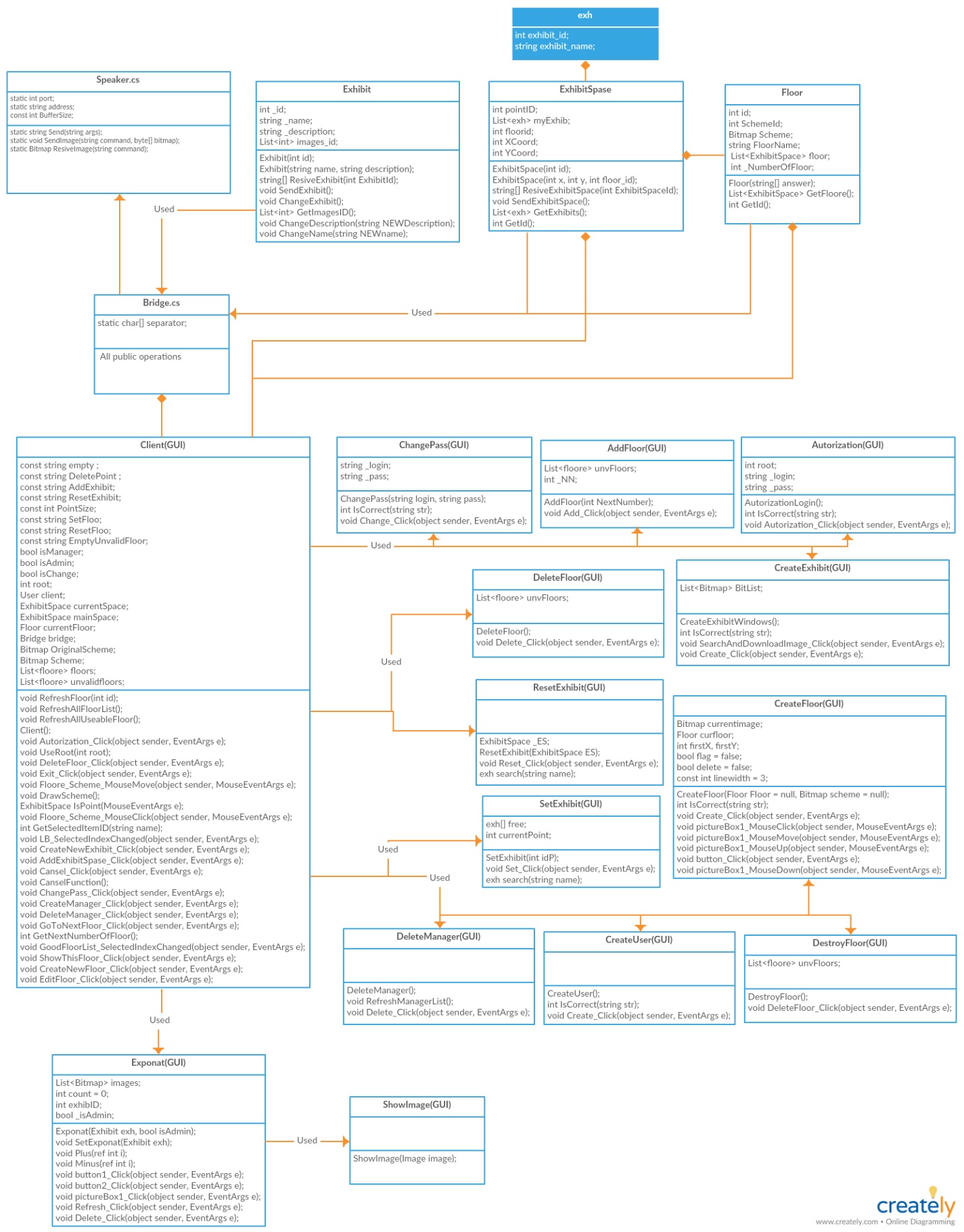


Рисунок 5

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 мультиплатформенность в будущем.