Отчет об индивидуальном проекте «ИС гостиничный комплекс»

Выполнен Темниковой Е. А.

г17210

ФИТ НГУ 2020г

Дисциплина «Базы данных»

Введение.

Проектное задание заключалось в том, чтобы разработать информационную систему гостиничного комплекса, работающую с базой данных и предоставляющую пользовательский интерфейс. Должны быть разработаны структура базы данных и приложение в архитектуре клиент-сервер на языке Java. Приложение должно обеспечивать выполнение операций внесения данных в базу данных, редактирования существующих данных и выполнение запросов, указанных в задании. Такая система может быть полезна сотрудникам гостиниц разных должностей, так как быстро предоставляет актуальные данные по запросу и имеет простой понятный интерфейс для пользователя. Система может быть использована для получения текущей информации о корпусах, номерах, постояльцах, а также её корректировка и внесение новых данных.

Цель проектной работы: разработать информационную систему гостиничного комплекса.

Задачи проектной работы:

1. Анализ проекта. Описываются обзор и анализ различных работ по проекту, формулируются основные результаты.
2. Проектирование системы. Описываются основные проектные решения работы
3. Реализация системы.
   1. Разработка базы данных и наполнение корректными данными.
   2. Разработать многооконный интерфейс пользователя и PL/SQL программы, реализующие алгоритмы, вносимые на серверную сторону приложения.
4. Произвести отладку и тестирование приложения.

Глава 1. Анализ проекта.

В данной главе проводится обзор и анализ различных работ по проекты, формулируются основные результаты.

**Основные сущности и отношения.**

В данном разделе приводится описание ключевых сущностей и связей между ними. Представляется модель данных, описывающая концептуальную схему предметной области проекта.

Всего сущностей : 8

*Гостиничный комплекс.*

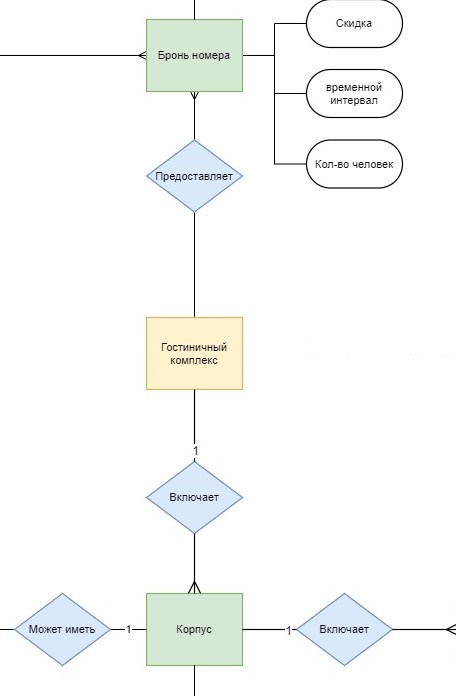


Рис. 1. Сущность «Гостиничный комплекс» и её связи в ER-диаграмме.

Главная сущность, не имеет собственных атрибутов, однако связывает все остальные сущности в единую систему. Имеет связь «предоставляет» с сущностью «бронь номера» и связь «включает» с сущностью «корпус».

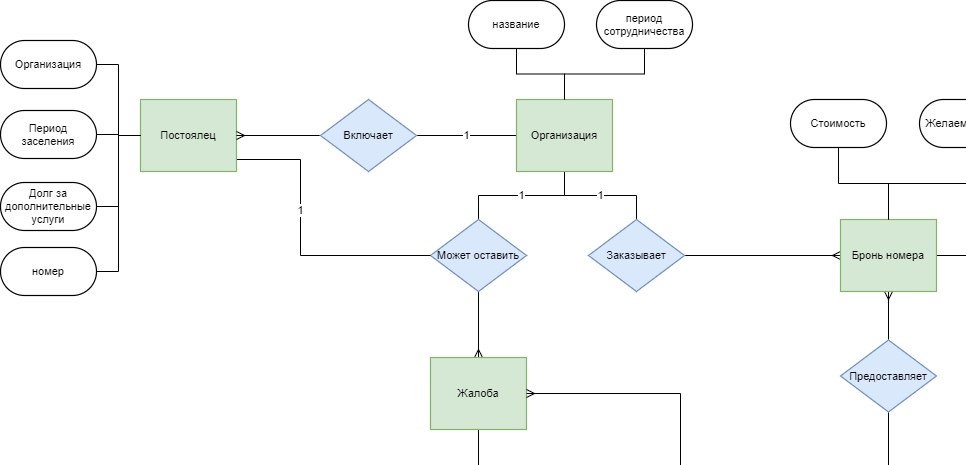
*Постоялец и организация*.

Рис. 2. Сущности «Постоялец» и «Организация» и их связи в ER-диаграмме.

Данные сущности имеют собственные атрибуты и связаны между собой отношением «один ко многим». Постоялец или организация могут оставлять жалобы на корпус, и поэтому имеют связь с сущностью «жалоба». Также от лица организации может быть осуществлена бронь номера, с сущностью «бронь» есть связь.

*Жалоба*.

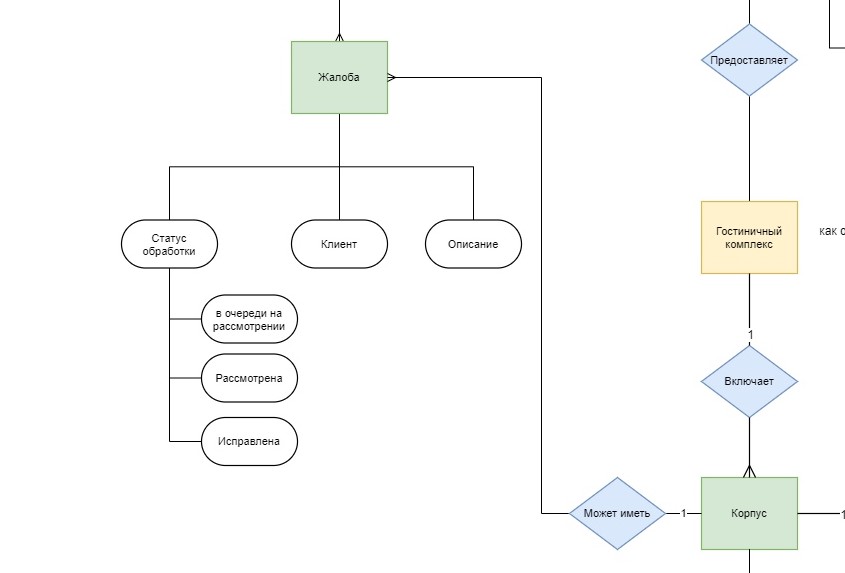


Рис. 3. Сущность «Жалоба» и её связи в ER-диаграмме.

Данная сущность имеет собственные атрибуты – клиент, описание и статус обработки, который разбивается на варианты атрибутов (в очереди, рассмотрена, исправлена). «Жалоба» имеет связь с «постоялец» и «организация», которые были рассмотрены выше, а также с «корпус». Одна организация или один постоялец могут оставлять несколько разных жалоб, а значит и один корпус может иметь несколько жалоб одновременно.

*Бронь.*

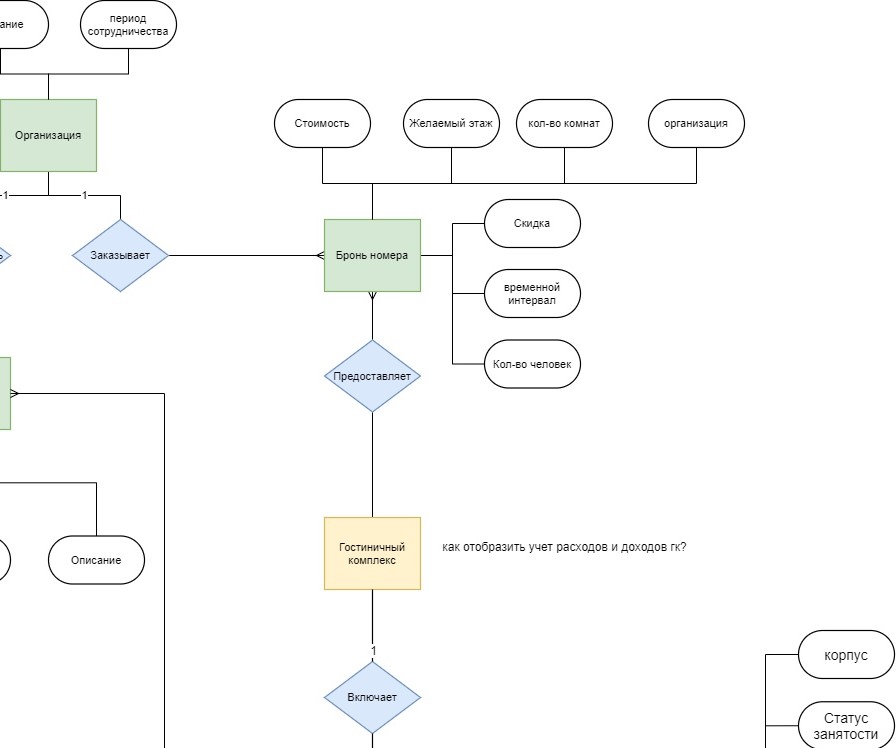
**

Рис. 4. Сущность «Бронь» и её связи в ER-диаграмме.

Данная сущность имеет множество собственных атрибутов. Каждая бронь принадлежит какой-то организации и предоставляется гостиничным комплексом.

*Корпус и номер.*

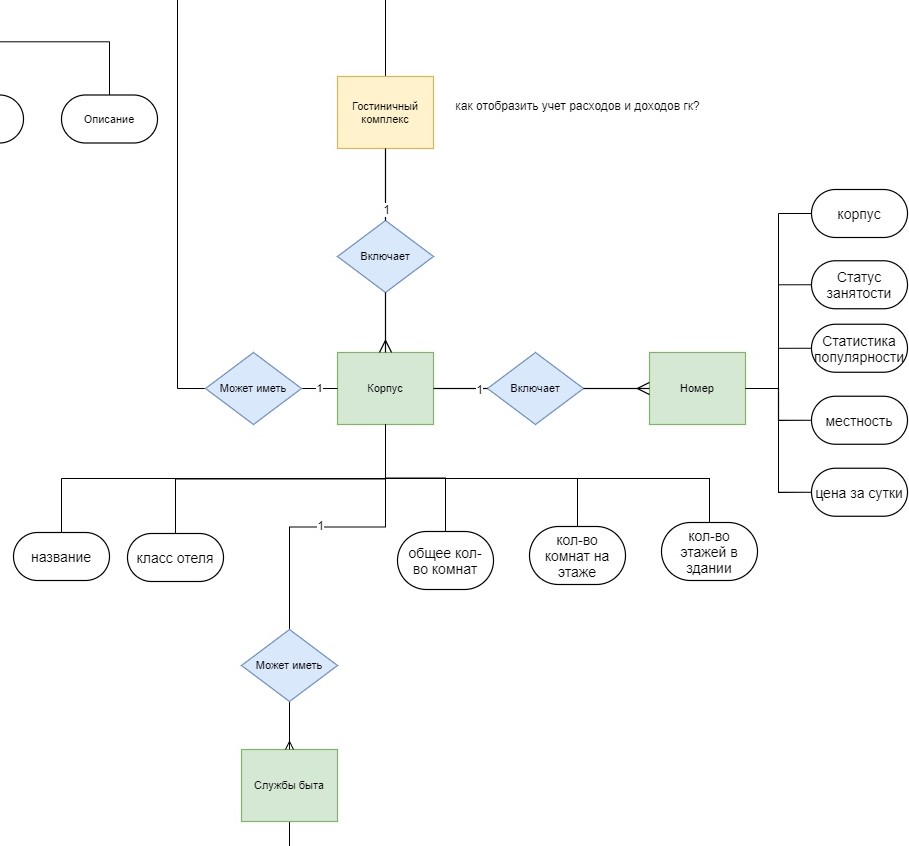


Рис. 5. Сущность «Бронь» и её связи в ER-диаграмме.

Данные сущности имеют свои атрибуты и связаны между собой отношением «один ко многим». Номер может иметь разный статус занятости (занят/свободен), местность (одноместный, двухместный и тд) и числиться только за одним корпусом. Корпус, как уже говорилось выше, может иметь несколько жалоб, а также каждый корпус включается в себя разные службы быта.

*Службы быта.*

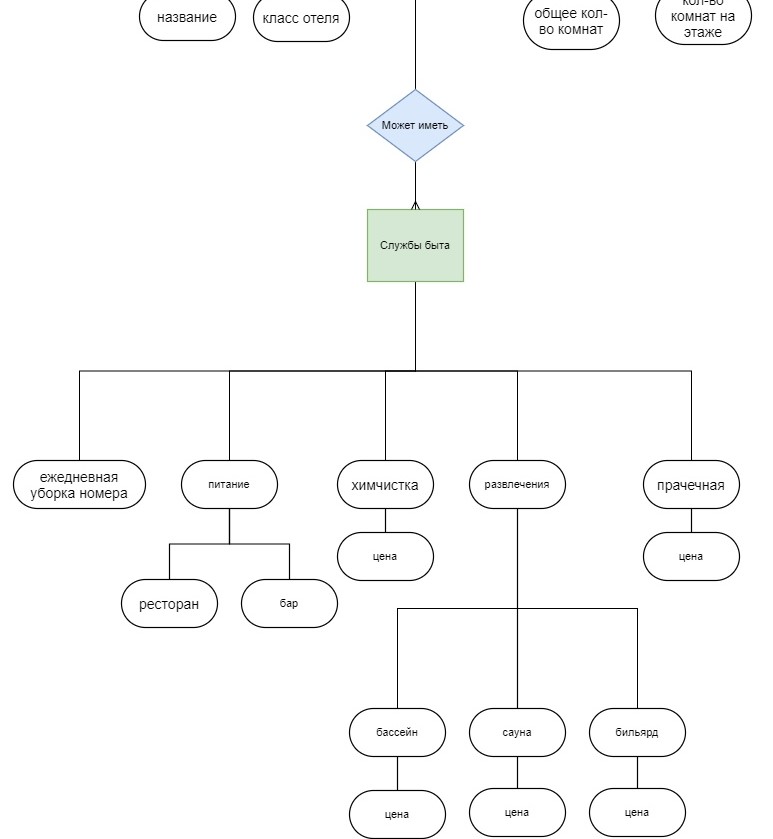
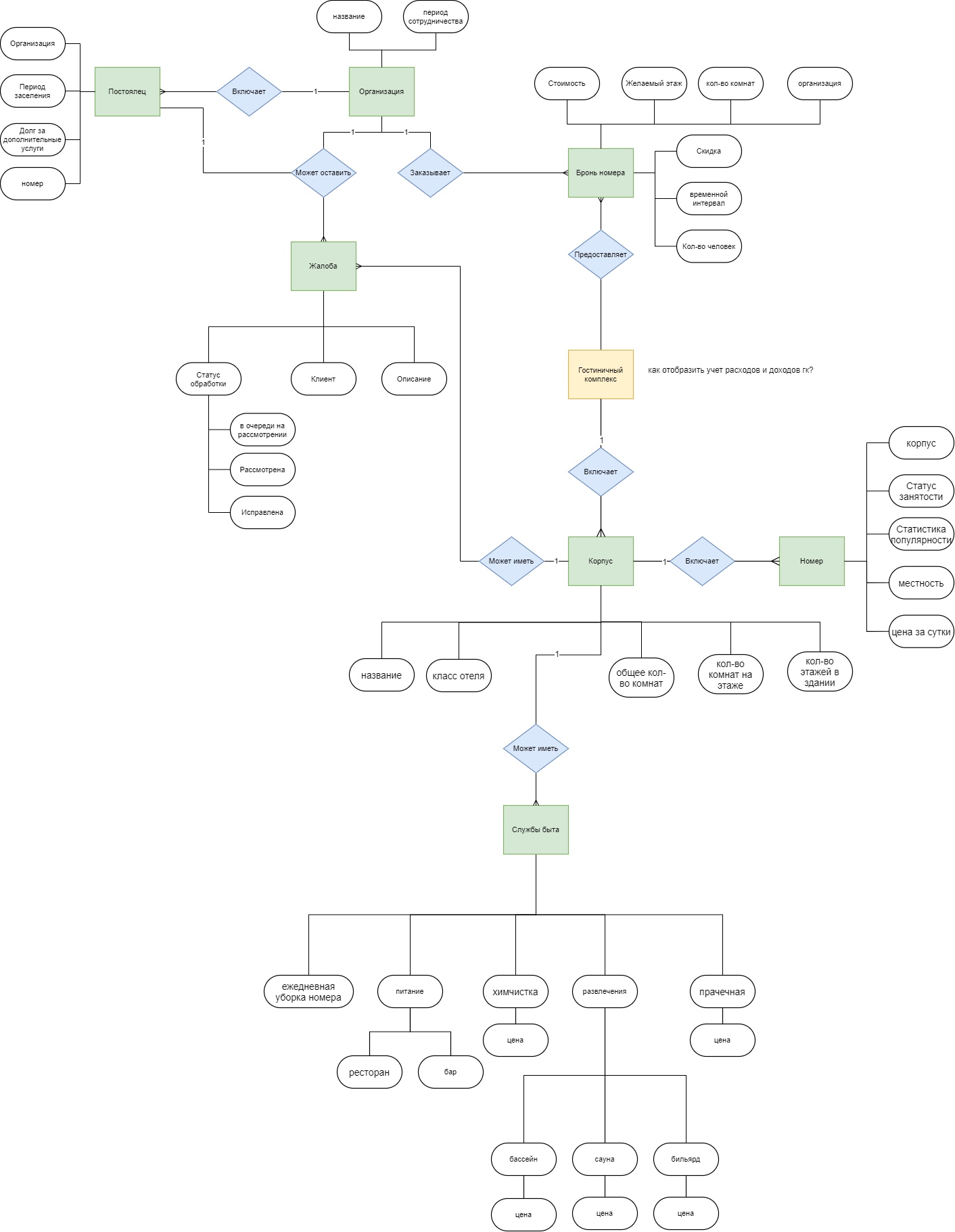


Рис. 6. Сущность «Службы быта» и её связи в ER-диаграмме.

Данная сущность имеет множество своих атрибутов, которые в свою очередь разбиваются на другие атрибуты. Каждая служба быта обязательно имеет свою стоимость. Службы быта можно добавлять или удалять для каждого корпуса персонально

Полная ER-диаграмма :



**Обеспечение целостности данных.**

В данном разделе выявляются требования к обеспечению целостности данных, и основных путях обеспечения их выполнения.

1. Все вносимые новые данные и корректировка старых данных должны соответствовать заявленным типам в базе данных. Для этого в пользовательских формах представлены инструкции как правильно заполнять поля. Если пользователь все-таки вводит несоответствующие данные, то запрос на БД не производится.
2. Номер не может быть создан, если не закреплен ни за одним корпусом.
3. Жалоба не может быть создана, если не принадлежит ни одному постояльцу/организации или не закреплена ни за одним корпусом.
4. При удалении службы быта из дополнительных услуг корпуса не должен уменьшаться долговой счет клиента, который пользовался этими услугами.

**Основные роли пользователей.**

В данном разделе выявляются основные роли пользователей приложения и основные сценарии использования их взаимодействия с приложением.

Администратор корпуса:

- вносит и корректирует информацию о корпусах

- просматривает и меняет статус жалоб

- просматривает популярность номеров

- просматривает информацию об организациях и постояльцах

- просматривает статистику номеров и популярность номеров

Дежурный администратор:

- просматривает разные списки постояльцев, историю заселения

- просматривает информацию о конкретном постояльце

- просматривает информацию о свободных/занятых номерах

- принимает жалобы и меняет их статус

Агент по бронированию:

- создает и отменяет бронь номеров

- просматривает информацию об организации и постояльцах

- просматривает информацию о свободных/занятых номерах

Кассир:

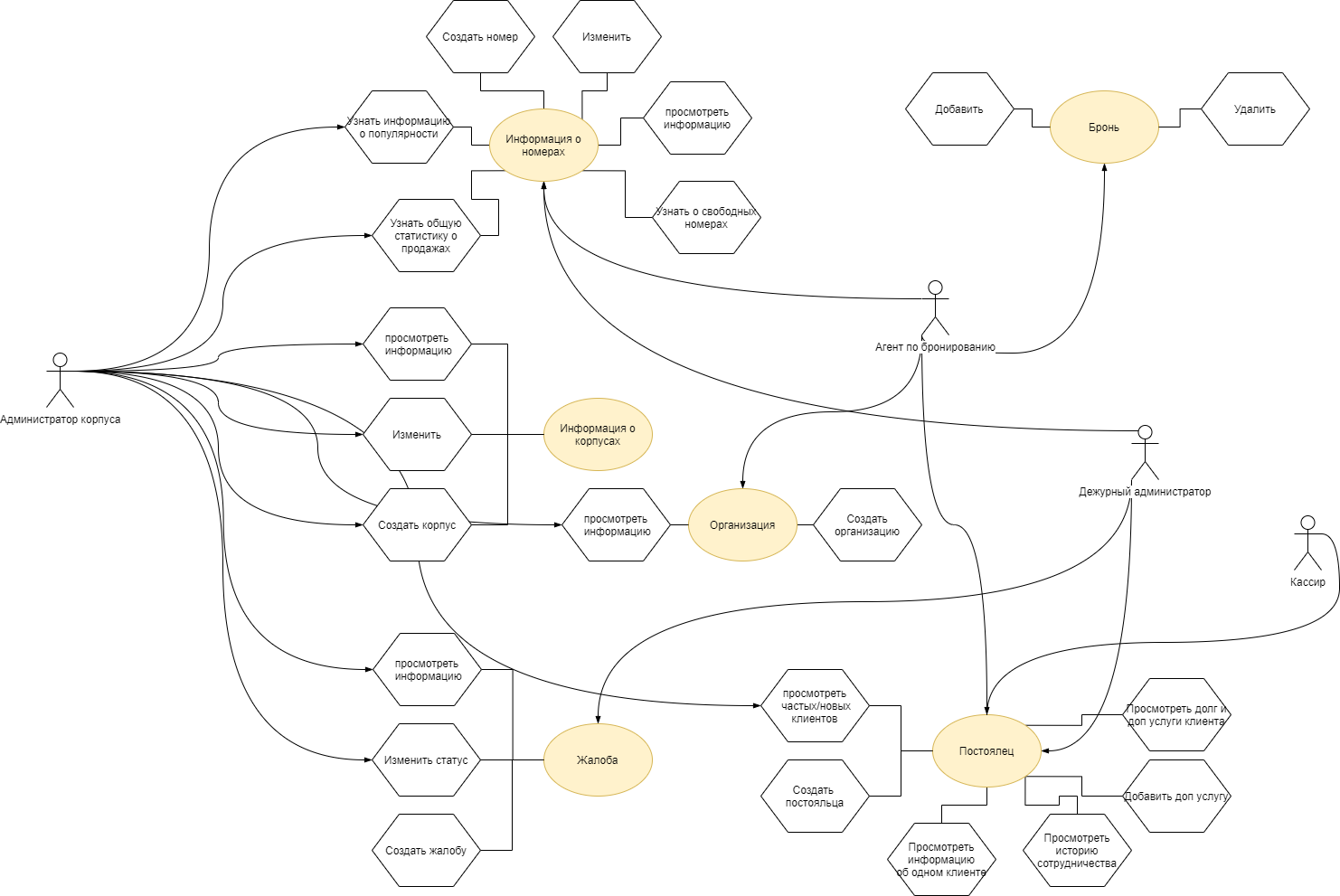
- просматривает всю информацию о постояльце

- рассчитывает клиента

- добавляет долг клиенту

**Диаграмма прецедентов.**

Данная диаграмма дает более общее представление о взаимодействиях пользователей с приложением. (изображение также есть в приложении к отчету)



Глава 2. Проектирование системы.

В данной главе описываются основные проектные решения работы приложения. Приводится общая архитектура приложения и ее основные части с описанием алгоритмов их взаимодействия. Также описываются проектные решения логического уровня, включая основные таблицы, способ представления в реляционной схеме всех типов сущностей. Приводится диаграмма схемы БД. Строятся алгоритмы обеспечения целостности данных, определяется разделение ответственности за обеспечение целостности между ядром СУБД, слоем хранимых процедур и алгоритмами клиентской части приложения. Раскрывается общее строение интерфейса в соответствии с ролями пользователей и прецедентами. Приводится таблица роли - прецеденты - формы с перечнем наиболее принципиальных форм интерфейса с кратким описанием функциональности каждой.

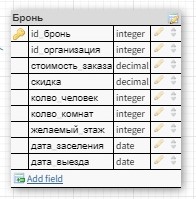
**Общая архитектура приложения.**

Приложение реализуется на строго типизированном объектно-ориентированном языке программирования Java с применением архитектурного шаблона «View-Controller pattern» и библиотеки для создания графического интерфейса «Swing». Контроллер в данном случае получает данные из БД, отдает их представлению в готовом для него виде. Представление принимает данные от контроллера и отображает их пользователю в графическом окне, также реагирует на действия пользователя, вновь отправляя запросы контроллеру.

Для удобства сопровождения кода, классы формируются в java-пакеты по принадлежности к определенному функционалу. Например, пакет «infCorpus» содержит все классы для работы формы «Информация о корпусах» и т.д. Также первоначально в меню пользователь выбирает свою роль в системе, и далее ему предлагается воспользоваться потенциально полезным для него функционалом.

**Логический уровень.**

Логическая модель базы данных включает в себя 8 таблиц с первичными и вторичными внешними ключами и ограничениями на данные.

***Бронь.***

Данная таблица имеет первичный ключ и один вторичный (id\_организации) с таблицей «Организация». Ограничения на остальные поля:

- стоимость заказа – неотрицательное вещественное число, не может быть null.

- скидка – неотрицательное вещественное число, не превышающее 100, может быть null.

- количество человек – неотрицательное целое число, не может быть null.

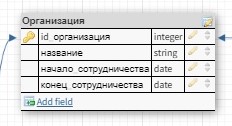
- количество комнат – неотрицательное целое число, может быть null.

- желаемый этаж – неотрицательное целое число, может быть null.

- дата заселения – тип Date, не позднее текущей даты при внесении, не может быть null.

-дата выезда – тип Date, не позднее текущей даты и даты заселения, не может быть null.

***Организация.***

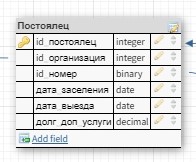
- id\_организация – первичный ключ.

- название – строка, длиной 50 символов, не может быть nulll.

- начало сотрудничества – тип Date, не может быть null.

- конец сотрудничества – тип Date, не может быть null.

***Постоялец.***



- id\_постоялец – первичный ключ.

- id\_организация – вторичный ключ с таблицей «Организация».

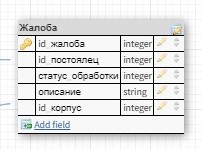
- id\_номер – вторичный ключ с таблицей «Номер».

- дата заселения– тип Date, не позднее текущей даты при внесении, не может быть null.

- дата выезда - тип Date, не позднее текущей даты и даты заселения, не может быть null.

- долг за доп. услуги - неотрицательное вещественное число, может быть null.

***Жалоба.***



- id\_жалоба – первичный ключ.

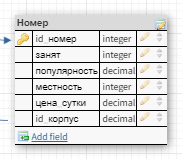
- id\_постоялец – вторичный ключ с таблицей «Постоялец».

- статус обработки – целочисленное число, принимающее значение 0, 1, 2. Что соответственно значит «на рассмотрении», «обрабатывается», «исправлено», не может быть null.

- описание – строка, длиною в 100 символов, не может быть null.

- id\_корпус – вторичный ключ с таблицей «Корпус».

***Номер.***



- id\_номер – первичный ключ.

- id\_корпус – вторичный ключ с таблицей «Корпус».

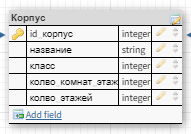
- занят – целочисленное число, принимающее значение 0 или 1, что соответственно значит «не занят», «занят», не может быть null.

- популярность – неотрицательное вещественное число, не превышающее 10, не может быть null.

- местность - целочисленное число, принимающее значение 1, 2, 3, 4, 5, не может быть null.

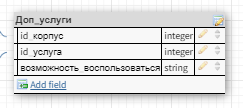
- цена в сутки - неотрицательное вещественное число, не может быть null.

***Корпус.***



* Id\_корпус – первичный ключ таблицы.
* Название – строка длиной не более 50 символов, не может быть null.
* Класс – целочисленное число, принимающее значение от 1 до 5, не может быть null.
* Количество комнат на этаже – неотрицательное целочисленное число, не может быть null.
* Количество этажей - неотрицательное целочисленное число, не может быть null.

***Дополнительные услуги.***



- id\_корпус – вторичный ключ с таблицей «Корпус».

- id\_услуга – вторичный ключ с таблицей «Услуга».

- возможность использования – строка, принимающая значения «yes» или «no» , не может быть null.

***Службы быта.***



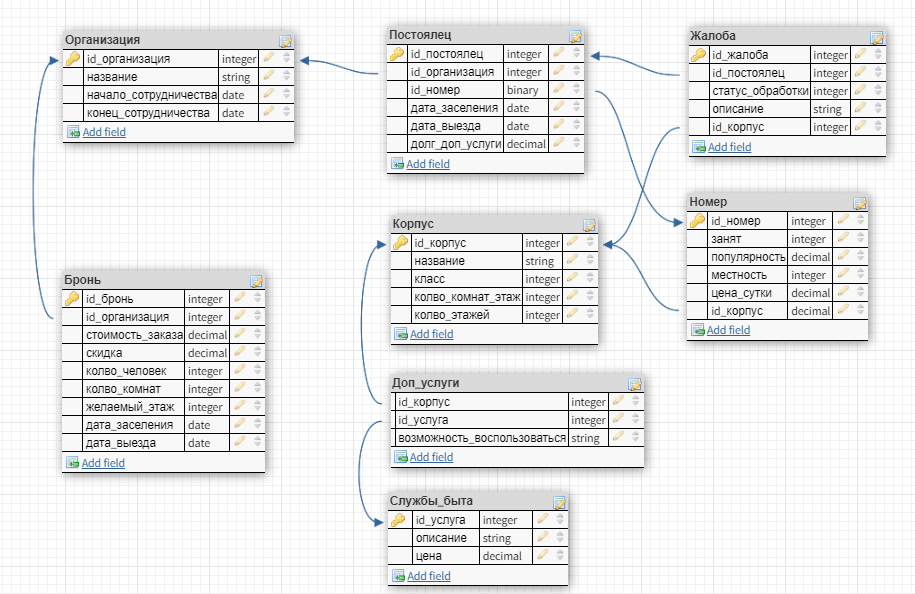
- id\_услуга – первичный ключ таблицы.

- описание – строка длиной не более 100 символов, не может быть null.

- цена – неотрицательное вещественное число, не может быть null.

**Диаграмма БД.**

Полная таблица с указанием связей, типов и первичных ключей.

****

**Алгоритмы обеспечения целостности данных.**

Целостность данных - важное свойство для базы данных, которое следует поддерживать на протяжении всей работы приложения. При правильном использовании оно обеспечивает корректность и валидность хранимых данных в любой момент времени. Также, с их помощью алгоритмов сохранения целостности данных можно обнаруживать ошибки в приложениях, которые тяжело найти другими способами.

В разработанной системе были использованы стандартные ограничения языка SQL :

PRIMARY KEY – уникальное поле, которое используется для однозначной идентификации записи в таблице. Для таблицы может быть создано только одно такое ограничение. Данное ограничение используется для обеспечения целостности сущности, которая описана таблицей.

FOREIGN KEY – ограничение защищает от действий, которые могут нарушить связи между таблицами, поддерживает ссылочную целостность данных.

CHECK – используется для ограничения множества значений, которые могут быть помещены в данный столбец. Это ограничение используется для обеспечения целостности предметной области, которую описывают таблицы в базе.

UNIQUE – ограничение обеспечивает отсутствие дубликатов в столбце или наборе столбцов.

**Строение интерфейса.**

Интерфейс стартует от основного меню, где пользователь выбирает роль, из которое он сможет просматривать или редактировать определенные части системы.

Далее пользователю открывается доступный для него функционал, отображенный в меню со списком форм. К этому меню можно возвращаться, после работы с формой и выбирать другие функции.

**Таблица прецедентов.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Роль | Прецедент | Форма | Описание |
| 1 | Администратор корпуса | Добавить / редактировать / просмотреть информацию о корпусах. | «Информация о корпусах» | Форма позволяет просмотреть информацию по всем существующим корпусам в БД, создать новый корпус, просмотреть доступные в корпусе дополнительные услуги, добавить или убрать дополнительную услугу в корпусе, изменить текущие данные о корпусе. |
| 2 | Администратор корпуса | Посмотреть текущие жалобы на корпус, изменить статус одной жалобы. | «Текущие жалобы» | Форма позволяет просмотреть весь список текущих жалоб о корпусе, просмотреть подробную информацию по конкретной жалобе, удалить или изменить её статус. |
| 3 | Администратор корпуса | Просмотреть популярность номеров. Посмотреть самые популярные номера. | «Популярность номеров» | Форма позволяет просмотреть топ-10 самых популярных номеров, а также просмотреть популярность конкретного номера. |
| 4 | Администратор корпуса | Просмотреть статистику о номерах. | «Статистика номеров» | Форма позволяет просмотреть объем продаж за указанный период, накладные расходы, процентное соотношение всех номеров к бронируемым. |
| 5 | Администратор корпуса | Посмотреть статистику по сотрудничеству с организациями. | «Организации» | Форма позволяет посмотреть общую статистику по фирмам за определенные период времени/за все время, забронировавших номера не менее указанного числа, посмотреть статистику по конкретной организации, узнать их предпочтения и дату последнего сотрудничества. |
| 6 | Администратор корпуса | Узнать самых частых гостей и новых клиентов. | «Частые гости и новые клиенты» | Форма позволяет увидеть наиболее частых посетителей за определенный период/за весь период в отдельном корпусе/по всем корпусам. Просмотреть список новых клиентов. |
| 7 | Дежурный администратор | Посмотреть текущие жалобы на корпус, изменить статус одной жалобы. | «Текущие жалобы» | Форма позволяет просмотреть весь список текущих жалоб о корпусе, просмотреть подробную информацию по конкретной жалобе, удалить или изменить её статус. |
| 8 | Дежурный администратор | Добавить новую жалобу. | «Добавить жалобу» | Форма позволяет принять новую жалобу от клиента. |
| 9 | Дежурный администратор | Узнать о занятости номеров. | «Свободные/занятые номера» | Форма позволяет узнать список свободных номеров на данный момент, узнать какие номера освободятся к определенной дате или узнать про занятость конкретного номера. |
| 10 | Дежурный администратор | Узнать всю информацию о постояльце или внести нового. | «Постоялец» | Форма позволяет найти всю информацию о существующем клиенте или добавить нового. Можно просмотреть историю сотрудничества, добавить в счет дополнительную услугу или просмотреть все услуги, которыми пользовался клиент. |
| 11 | Дежурный администратор | Узнать список заселявшихся постояльцев. | «История заселений» | Форма позволяет просмотреть историю заселений по определенному корпусу за конкретный период. |
| 12 | Дежурный администратор | Узнать самых частых гостей и новых клиентов. | «Частые гости и новые клиенты» | Форма позволяет увидеть наиболее частых посетителей за определенный период/за весь период в отдельном корпусе/по всем корпусам. Просмотреть список новых клиентов. |
| 13 | Агент по бронированию | Получить или добавить информацию о брони. | «Бронь номеров» | Форма позволяет просмотреть список существующих броней, добавить новую или удалить/отменить старую. |
| 14 | Агент по бронированию | Узнать всю информацию о постояльце или внести нового. | «Постоялец» | Форма позволяет найти всю информацию о существующем клиенте или добавить нового. Можно просмотреть историю сотрудничества, добавить в счет дополнительную услугу или просмотреть все услуги, которыми пользовался клиент. |
| 15 | Агент по бронированию | Посмотреть статистику по сотрудничеству с организациями. | «Организации» | Форма позволяет посмотреть общую статистику по фирмам за определенные период времени/за все время, забронировавших номера не менее указанного числа, посмотреть статистику по конкретной организации, узнать их предпочтения и дату последнего сотрудничества. |
| 16 | Агент по бронированию | Просмотреть популярность номеров. Посмотреть самые популярные номера. | «Популярность номеров» | Форма позволяет просмотреть топ-10 самых популярных номеров, а также просмотреть популярность конкретного номера. |
| 17 | Агент по бронированию | Узнать о занятости номеров. | «Свободные/занятые номера» | Форма позволяет узнать список свободных номеров на данный момент, узнать какие номера освободятся к определенной дате или узнать про занятость конкретного номера. |
| 18 | Агент по бронированию | Узнать список заселявшихся постояльцев. | «История заселений» | Форма позволяет просмотреть историю заселений по определенному корпусу за конкретный период. |
| 19 | Кассир | Узнать всю информацию о постояльце или внести нового. | «Постоялец» | Форма позволяет найти всю информацию о существующем клиенте или добавить нового. Можно просмотреть историю сотрудничества, добавить в счет дополнительную услугу или просмотреть все услуги, которыми пользовался клиент. |

**Вопросы авторизации.**

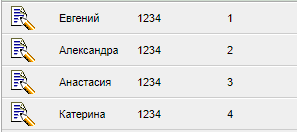
Для авторизации в базе данных хранится таблица с логинами и паролями пользователей, также хранится номер, показывающий роль данного пользователя:

1 – администратор корпусов

2 – дежурный администратор

3 – агент по бронированию

4 – кассир



После ввода пользователем логина и пароля таким образом определяется его роль.

Глава 3. Реализация системы

В данной главе описываются этапы и особенности реализации системы, приводится номенклатура SQL скриптов, описывается тестовый набор данных, скрипты обеспечения целостности данных, реализация пользовательского интерфейса, обработка ошибок и исключений.

**Общий объем работ по программированию.**

В соответствие с раннее описанной архитектурой были реализованы следующие пакеты:

- infCorpus – форма «Информация о корпусах»

- infRoom – форма «Информация о номере»

- booking – форма «Бронь»

- complaint – формы «Текущие жалобы» и «Добавить жалобу»

- historySettlement – форма «История заселений»

- organization – форма «Организации»

- popularRooms – форма «Популярные номера»

- guest – форма «Постоялец»

- occupancyRooms – форма «Занятость номеров»

- statisticsRooms – форма «Статистика номеров»

- importantGuest – формы «Частые гости» и «Новые гости»

В каждом из пакетов есть свой класс «StartMenu» (стартовое меню, с которого начинается работа с формой), «HelpNote» (подсказки для пользователя по формату вводимых данных), «Controller» (класс, который взаимодействует с БД, делает запросы и возвращает ответ). Также есть класс, не вошедший в отдельный пакет – «Menu», который устанавливает соединение с БД и отрисовывает меню с выбором роли пользователя.

Для работы с БД использовался драйвер JDBC, который решает следующие задачи:

* Создание соединения с БД.
* Создание SQL выражений.
* Выполнение SQL – запросов.
* Просмотр и модификация полученных записей.

Код подключения с использованием драйвера:



Для реализации пользовательского интерфейса использовалась библиотека «swing». (подробнее в разделе «реализация пользовательского интерфейса»)

**Номенклатура SQL скриптов.**

Таблицы создавались с помощью команды «Create table» с описанием основных ограничений («Primary key», «Foreign key»).

CREATE TABLE «имя таблицы» (  
    поле1 тип [NOT NULL],  
    . . .  
    CONSTRAINT PK\_название  PRIMARY KEY (поле1),  
    CONSTRAINT FK\_название FOREIGN KEY (поле1)  
    REFERENCES Таблица(поле1)  
);

Пример:

CREATE TABLE "БРОНЬ"

( "ID\_БРОНЬ" NUMBER(10,0) NOT NULL ENABLE,

"ID\_ОРГАНИЗАЦИЯ" NUMBER(10,0) NOT NULL ENABLE,

"СТОИМ\_ЗАКАЗА" NUMBER(10,2) NOT NULL ENABLE,

"СКИДКА" NUMBER(4,2),

"КОЛВО\_ЧЕЛ" NUMBER(3,0) NOT NULL ENABLE,

"КОЛВО\_КОМНАТ" NUMBER(2,0),

"ЖЕЛАЕМЫЙ\_ЭТАЖ" NUMBER(2,0),

"ДАТА\_ЗАСЕЛ" DATE NOT NULL ENABLE,

"ДАТА\_ВЫЕЗДА" DATE NOT NULL ENABLE,

CONSTRAINT "БРОНЬ\_PK" PRIMARY KEY ("ID\_БРОНЬ") ENABLE,

CONSTRAINT "ОРГ\_БРОНЬ\_FK" FOREIGN KEY ("ID\_ОРГАНИЗАЦИЯ")

REFERENCES "ОРГАНИЗАЦИЯ" ("ID\_ОРГАНИЗАЦИЯ") ENABLE

);

Данные вводились с помощью команды «INSERT INTO»:

INSERT INTO таблица  
VALUES (столбец1, столбец2 … );

Пример:

INSERT INTO КОРПУС  
VALUES (1, ‘Золотая долина’, 5, 10, 5);

**Характеристики тестового набора данных.**

Необходимые характеристики для тестового набора данных, который будет использоваться в таблицах БД для тестирования системы:

* данные должны обеспечивать реальное тестирование системы во всех режимах
* все данные по содержанию должны соответствовать названиям соответствующих полей таблиц БД
* не могут быть заполнены формально, должны отражать примеры возможных реальных данных
* объем и содержание тестовых данных должны быть достаточным для тестирования работы приложения в различных режимах
* в таблицах должна быть информация не менее чем о пяти – семи объектах каждого вида

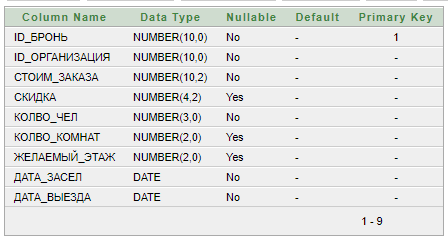
Пример sql кода для ввода тестовых данных:

INSERT INTO КОРПУС  
VALUES (*1, ‘Золотая долина’, 5, 10, 5*);

**Обеспечение целостности данных.**

В базе данных были введены ограничения на ввод данных в таблицу по типу данных, некоторые данные должны являться внешними ключами. Данные, на которые ссылаются другие ключи не могут быть удалены, либо применяется алгоритм каскадного удаления. Некоторые поля не могут быть null. Ограничения продумывались отдельно для каждой таблицы.

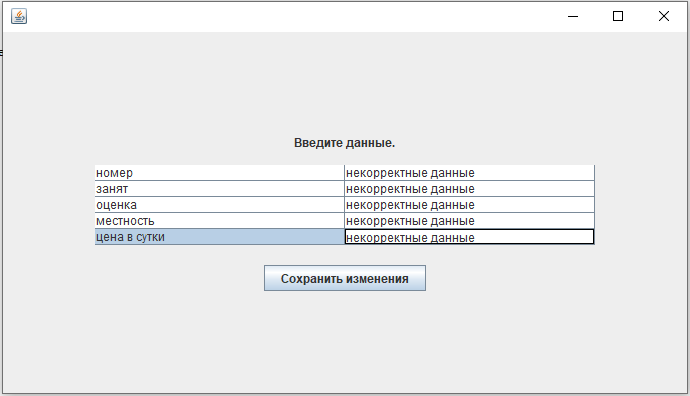
Пример одной из таблиц (Бронь):

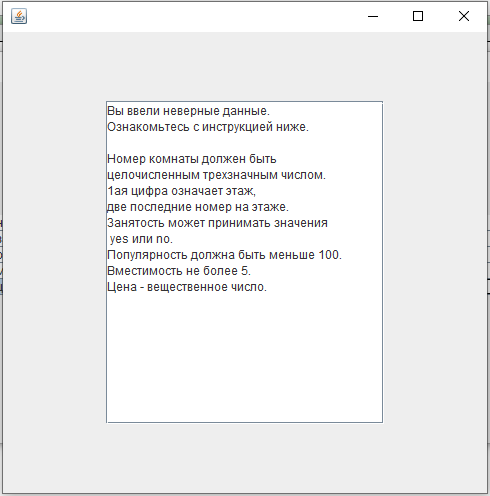


Здесь id\_организация есть внешний ключ на таблицу «Организация», соответственно он не может быть пустым. Такое ограничение обосновано, так как бронировать номера от несуществующей или пустой организации бессмысленно, каждая бронь должна иметь организатора.

Далее, со стороны клиентской части также тщательно проверяется ввод пользователя. Если пользователь не ввел данных или ввел в некорректном формате, выдается предупреждающее окно и предлагается ввести данные снова.

Пример:



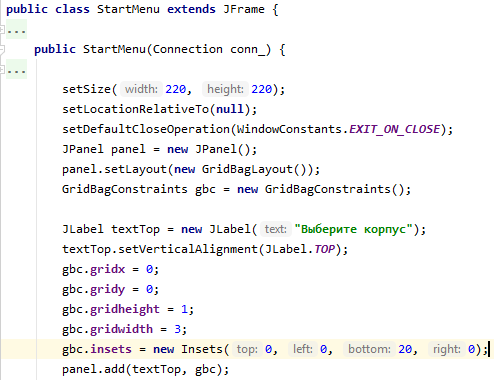


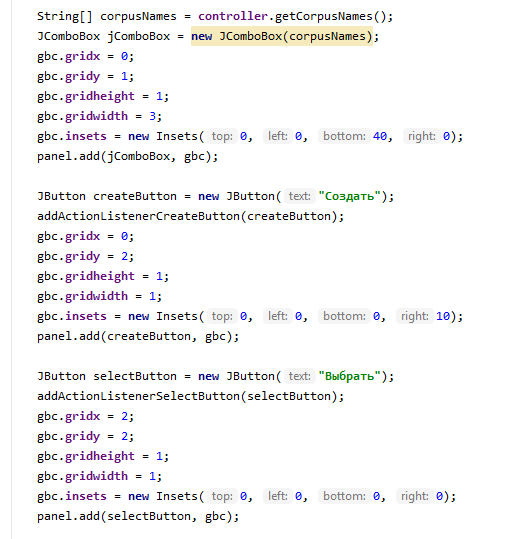
**Реализация пользовательского интерфейса системы.**

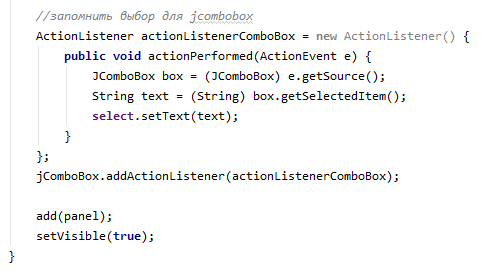
Для реализации пользовательского интерфейса системы использовалась графическая библиотека компонентов «Swing». Swing позволяет легко создавать новые компоненты, наследуясь от существующих, и поддерживает различные стили.

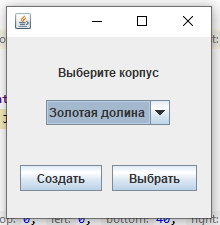
Пример использования компонентов

Swing в коде:









С помощью этой библиотеки были разработаны формы с доступным пользователю функционалом, подсказки по заполнению этих форм, обработка пользовательской реакции.

**Обработка ошибок и исключений.**

При использовании приложения могут быть ошибки следующего рода:

1. Ошибки пользователя - неправильный ввод или редактирование данных.
2. Ошибки программного кода, возникающие при неучтенном сценарии работы в тестировании.
3. Ошибки на серверной стороне, отказ в соединении.

Для ошибок первого типа предусмотрены диалоговые окна, подсказывающие пользователю как правильно вводить данные, какого они должны быть типа и формата. Для ошибок второго и третьего типа были использованы в программном коде операторы try/catch, с выводом описания ошибки и показом диалогового окна.

Глава 4. Тестирование.

//к данной главе будут добавлены скрины в каждый раздел

В данной главе описываются проведенные мероприятия и способы тестирования приложения. Было произведено функциональное тестирование, показывающее что система выполняет свои функции корректно, и нефункциональное, показывающее что система выполняет свои функции в срок, в должном объеме и с приемлемым качеством, и пользоваться ею удобно и надежно.

**Критерии для набора данных.**

Тестовый набор данных был составлен в достаточном объеме, загружен в базу данных и полностью соответствовал данным критериям:

* обеспечивает реальное тестирование системы во всех режимах
* все данные по содержанию должны соответствуют названиям соответствующих полей таблиц БД
* отражают примеры возможных реальных данных
* объем и содержание тестовых данных достаточны для тестирования работы приложения в различных режимах
* в таблицах имеется информация не менее чем о пяти – семи объектах каждого вида

Пример (таблица Номер):



**Порядок работ по тестированию приложения.**

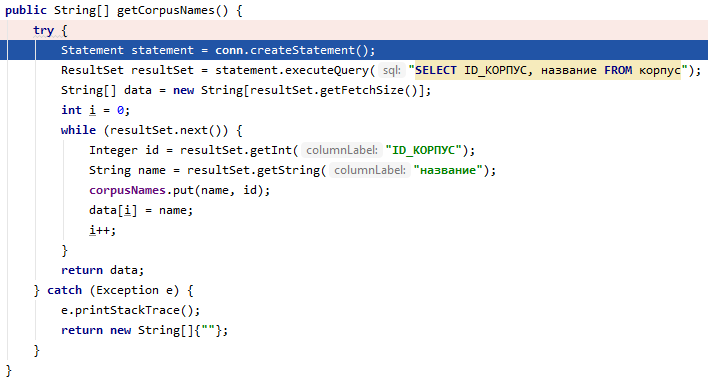
Были проведены следующие виды функционального тестирования:

*Компонентное тестирование* – испытание отдельных программных компонентов информационной системы, в ходе которых подтверждается корректность проводимых этими компонентами вычислений. Например, работа одной формы, прием данных, проверка на корректность, запросы в БД.

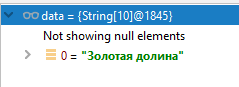
Пример:

Проверка того, что корректно выполняется запрос в Бд на получение списка названий корпусов.

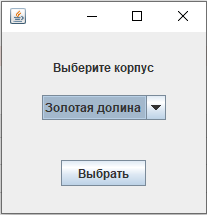
Тестируемый код:



Принятые данные:



Отрисовка окна:



*Интеграционное тестирование* – испытания, направленные на выявление проблем взаимодействия отдельных компонентов системы. Например, взаимодействие разных пакетов, переключение форм.

Проверка меню на вызов разных форм из разных пакетов по запросу пользователя.

Тестируемый класс - Menu

Ожидаемое поведение – быстрое взаимодействие с другими классами

*Тестирование прототипа* – испытания информационной системы на [первых этапах ее разработки](http://itconcord.ru/articles/stages/), когда готовы не все ее функциональные блоки. Отсутствующие компоненты заменяются функциональными заглушками, имитирующими их будущую работу. Например, проводилось тестирование интерфейса и отрисовки всех компонентов в правильных масштабах без работы контроллера.

Также были проведены следующие виды нефункционального тестирования:

Испытания *пользовательского интерфейса* на предмет удобства и скорости отклика на действия пользователя.

Испытания на надежность *вводимых в систему данных*. Данные должны проверяться по формату и типу, при неправильном вводе не должны выполняться запросы в БД.

**Результаты.**

Были проведены функциональные и нефункциональные тесты, покрывающие всю систему как покомпонентно, так и интеграционно. Были проверены обработка ошибок и исключений, удобство и доступность многооконного интерфейса, корректность работы с БД. Система предоставляет заявленный функционал, поддерживает возможность использования в нескольких ролях. База данных разработана в соответствии со схемой данных, наполнена корректными данными.

Заключение.

В результате всей работы была разработана информационная система гостиничного комплекса, работающая с базой данных и предоставляющая многооконный пользовательский интерфейс.

Был проведен анализ проекта, в котором были описаны основные сущности и отношения.

В ходе работы была спроектирована и разработана база данных, инфологическая модель, также БД была наполнена корректными тестовыми данными. Был смоделирован и разработан пользовательский интерфейс, а также PL/SQL программы, реализующие алгоритмы, вносимые изменения на серверную сторону приложения.

Также была произведена отладка и полное тестирование приложения.

Были получены навыки подключения к БД через драйвер djbc, проектирования и моделирования БД и приложения, программной реализации запросов на БД.

Литература.

[Основы языка SQL/PL](https://oracle-patches.com/db/sql/3125-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0-pl-sql)

[Краткий справочник по Oracle7 и Pl/SQL](http://www.codenet.ru/db/oracle/ora1/)

[Он-лайн учебник и справочные материалы по PL/SQL](https://oracleplsql.ru/contents-oracle-plsql.html)

[Руководство по JDBC](https://proselyte.net/tutorials/jdbc/introduction/)

[Подключение к базам данных Oracle](https://oracle-patches.com/coding/3042-java-%D0%BF%D0%BE%D0%B4%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BA-%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-oracl)

Приложение.

Скрипты sql делятся на разделы:

* создание таблиц
* наполнение БД
* запросы

Создание таблиц.

CREATE TABLE "БРОНЬ"

( "ID\_БРОНЬ" NUMBER(10,0) NOT NULL ENABLE,

"ID\_ОРГАНИЗАЦИЯ" NUMBER(10,0) NOT NULL ENABLE,

"СТОИМ\_ЗАКАЗА" NUMBER(10,2) NOT NULL ENABLE,

"СКИДКА" NUMBER(4,2),

"КОЛВО\_ЧЕЛ" NUMBER(3,0) NOT NULL ENABLE,

"КОЛВО\_КОМНАТ" NUMBER(2,0),

"ЖЕЛАЕМЫЙ\_ЭТАЖ" NUMBER(2,0),

"ДАТА\_ЗАСЕЛ" DATE NOT NULL ENABLE,

"ДАТА\_ВЫЕЗДА" DATE NOT NULL ENABLE,

CONSTRAINT "БРОНЬ\_PK" PRIMARY KEY ("ID\_БРОНЬ") ENABLE,

CONSTRAINT "ОРГ\_БРОНЬ\_FK" FOREIGN KEY ("ID\_ОРГАНИЗАЦИЯ")

REFERENCES "ОРГАНИЗАЦИЯ" ("ID\_ОРГАНИЗАЦИЯ") ENABLE

)

CREATE TABLE "ДОП\_УСЛУГИ"

( "ID\_КОРПУС" NUMBER(10,0) NOT NULL ENABLE,

"ID\_УСЛУГА" NUMBER(10,0) NOT NULL ENABLE,

"ВОЗ\_ПОЛ" VARCHAR2(3) NOT NULL ENABLE,

CONSTRAINT "КОР\_ДОП\_FK" FOREIGN KEY ("ID\_КОРПУС")

REFERENCES "КОРПУС" ("ID\_КОРПУС") ENABLE,

CONSTRAINT "УС\_ДОП\_FK" FOREIGN KEY ("ID\_УСЛУГА")

REFERENCES "СЛУЖБЫ\_БЫТА" ("ID\_УСЛУГА") ENABLE

)

CREATE TABLE "ЖАЛОБА"

( "ID\_ЖАЛОБА" NUMBER(10,0) NOT NULL ENABLE,

"ID\_ПОСТОЯЛЕЦ" NUMBER(10,0) NOT NULL ENABLE,

"ID\_КОРПУС" NUMBER(10,0) NOT NULL ENABLE,

"СТАТУС\_ОБР" NUMBER(1,0) NOT NULL ENABLE,

"ОПИСАНИЕ" VARCHAR2(50) NOT NULL ENABLE,

CONSTRAINT "ЖАЛОБА\_PK" PRIMARY KEY ("ID\_ЖАЛОБА") ENABLE,

CONSTRAINT "ПОСТ\_ЖАЛОБА\_FK" FOREIGN KEY ("ID\_ПОСТОЯЛЕЦ")

REFERENCES "ПОСТОЯЛЕЦ" ("ID\_ПОСТОЯЛЕЦ") ENABLE,

CONSTRAINT "КОР\_ЖАЛ\_FK" FOREIGN KEY ("ID\_КОРПУС")

REFERENCES "КОРПУС" ("ID\_КОРПУС") ENABLE

)

CREATE TABLE "ИСТОРИЯ"

( "ID\_ЗАПИСЬ" NUMBER NOT NULL ENABLE,

"ID\_ПОСТОЯЛЕЦ" NUMBER NOT NULL ENABLE,

"ID\_КОРПУС" NUMBER NOT NULL ENABLE,

"ID\_НОМЕР" NUMBER NOT NULL ENABLE,

"ДАТА\_ЗАСЕЛ" DATE,

"ДАТА\_ВЫЕЗД" DATE,

"ДОЛГ" NUMBER,

CONSTRAINT "ИСТОРИЯ\_PK" PRIMARY KEY ("ID\_ЗАПИСЬ") ENABLE,

CONSTRAINT "ИСТОРИЯ\_FK" FOREIGN KEY ("ID\_ПОСТОЯЛЕЦ")

REFERENCES "ПОСТОЯЛЕЦ" ("ID\_ПОСТОЯЛЕЦ") ENABLE,

CONSTRAINT "ИСТОРИЯ\_FK2" FOREIGN KEY ("ID\_НОМЕР")

REFERENCES "НОМЕР" ("ID\_НОМЕР") ENABLE,

CONSTRAINT "ИСТОРИЯ\_FK3" FOREIGN KEY ("ID\_КОРПУС")

REFERENCES "КОРПУС" ("ID\_КОРПУС") ENABLE

)

CREATE OR REPLACE TRIGGER "BI\_ИСТОРИЯ"

before insert on "ИСТОРИЯ"

for each row

begin

select "ИСТОРИЯ\_SEQ".nextval into :NEW.ID\_ЗАПИСЬ from dual;

end;

ALTER TRIGGER "BI\_ИСТОРИЯ" ENABLE

CREATE TABLE "ИСТОРИЯ\_УСЛУГ"

( "ID\_ЗАПИСЬ" NUMBER,

"ID\_УСЛУГИ" NUMBER,

"КОЛИЧЕСТВО" NUMBER,

CONSTRAINT "ИСТОРИЯ\_УСЛУГ\_FK" FOREIGN KEY ("ID\_ЗАПИСЬ")

REFERENCES "ИСТОРИЯ" ("ID\_ЗАПИСЬ") ENABLE,

CONSTRAINT "ИСТОРИЯ\_УСЛУГ\_FK2" FOREIGN KEY ("ID\_УСЛУГИ")

REFERENCES "СЛУЖБЫ\_БЫТА" ("ID\_УСЛУГА") ENABLE

)

CREATE TABLE "КОРПУС"

( "ID\_КОРПУС" NUMBER(10,0) NOT NULL ENABLE,

"НАЗВАНИЕ" VARCHAR2(50) NOT NULL ENABLE,

"КЛАСС" NUMBER(1,0) NOT NULL ENABLE,

"КОМН\_ЭТАЖ" NUMBER(2,0) NOT NULL ENABLE,

"КОЛВО\_ЭТАЖЕЙ" NUMBER(2,0) NOT NULL ENABLE,

CONSTRAINT "КОРПУС\_PK" PRIMARY KEY ("ID\_КОРПУС") ENABLE

)

CREATE TABLE "НОМЕР"

( "ID\_НОМЕР" NUMBER(10,0) NOT NULL ENABLE,

"ID\_КОРПУС" NUMBER(10,0) NOT NULL ENABLE,

"ЗАНЯТ" VARCHAR2(3) NOT NULL ENABLE,

"ПОПУЛЯРНОСТЬ" NUMBER(5,2),

"МЕСТНОСТЬ" NUMBER(1,0) NOT NULL ENABLE,

"ЦЕНА\_СУТКИ" NUMBER(6,2) NOT NULL ENABLE,

CONSTRAINT "НОМЕР\_PK" PRIMARY KEY ("ID\_НОМЕР") ENABLE,

CONSTRAINT "КОР\_НОМ\_FK" FOREIGN KEY ("ID\_КОРПУС")

REFERENCES "КОРПУС" ("ID\_КОРПУС") ENABLE

)

CREATE TABLE "ОРГАНИЗАЦИЯ"

( "ID\_ОРГАНИЗАЦИЯ" NUMBER(10,0) NOT NULL ENABLE,

"НАЗВАНИЕ" VARCHAR2(50) NOT NULL ENABLE,

"НАЧ\_СОТР" DATE NOT NULL ENABLE,

"КОН\_СОТР" DATE,

CONSTRAINT "ОРГАНИЗАЦИЯ\_PK" PRIMARY KEY ("ID\_ОРГАНИЗАЦИЯ") ENABLE

)

CREATE TABLE "ПОСТОЯЛЕЦ"

( "ID\_ПОСТОЯЛЕЦ" NUMBER(10,0) NOT NULL ENABLE,

"ID\_ОРГАНИЗАЦИЯ" NUMBER(10,0) NOT NULL ENABLE,

"NAME" VARCHAR2(4000) NOT NULL ENABLE,

CONSTRAINT "ПОСТОЯЛЕЦ\_PK" PRIMARY KEY ("ID\_ПОСТОЯЛЕЦ") ENABLE,

CONSTRAINT "ОРГ\_ПОСТ\_FK" FOREIGN KEY ("ID\_ОРГАНИЗАЦИЯ")

REFERENCES "ОРГАНИЗАЦИЯ" ("ID\_ОРГАНИЗАЦИЯ") ENABLE

)

CREATE TABLE "СЛУЖБЫ\_БЫТА"

( "ID\_УСЛУГА" NUMBER(10,0) NOT NULL ENABLE,

"ОПИСАНИЕ" VARCHAR2(100) NOT NULL ENABLE,

"ЦЕНА" NUMBER(6,2) NOT NULL ENABLE,

CONSTRAINT "УСЛУГА\_PK" PRIMARY KEY ("ID\_УСЛУГА") ENABLE

)

Наполнение БД.

insert into БРОНЬ values (4,2,1000,10,2,1,1,date'2020-05-31',date'2020-06-01')

insert into КОРПУС values (2, 'Удача', 3, 5,4)

insert into КОРПУС values (3, 'Барышни', 3, 5,4)

insert into КОРПУС values (4, 'Цветок', 3, 5,4)

insert into КОРПУС values (4, 'Радость', 3, 5,4)

insert into КОРПУС values (5, 'Солнце', 3, 5,4)

insert into ДОП\_УСЛУГИ values (2,1,'yes')

insert into ДОП\_УСЛУГИ values (2,2,'yes')

insert into ДОП\_УСЛУГИ values (2,3,'yes')

insert into ПОСТОЯЛЕЦ values (2,2,'Кирилл Порохов')

insert into ПОСТОЯЛЕЦ values (3,4,'Илья Петров')

insert into ПОСТОЯЛЕЦ values (4,4,'Мария Петрова')

Запросы.

**SELECT ID\_КОРПУС, название FROM корпус**

**-------------------------------------------------------------------------------------------**

**select distinct id\_постоялец from история where (ДАТА\_ЗАСЕЛ between sysdate-7 and sysdate) and id\_корпус=**id

**-------------------------------------------------------------------------------------------select distinct name from постоялец where id\_постоялец=**id

**-------------------------------------------------------------------------------------------select distinct id\_организация from постоялец where id\_постоялец=**id

**-------------------------------------------------------------------------------------------select distinct название from организация where id\_организация=**id

**-------------------------------------------------------------------------------------------**

**select id\_постоялец,** *count***(id\_постоялец) from история where (ID\_КОРПУС=id and (rownum < 11) group by id\_постоялец order by** *count***(id\_постоялец) desc**;

**-------------------------------------------------------------------------------------------**

**select id\_постоялец,** *count***(id\_постоялец) from история where (ID\_КОРПУС=** id**) and (rownum < 11) and ((date'date\_start'<= ДАТА\_ЗАСЕЛ and date'date\_**finish**' >= ДАТА\_ВЫЕЗД) or (date'date\_start'** **between ДАТА\_ЗАСЕЛ and ДАТА\_ВЫЕЗД) or (date'"** + finish + **"' between ДАТА\_ЗАСЕЛ and ДАТА\_ВЫЕЗД)) group by id\_постоялец order by** *count***(id\_постоялец) desc**

**-------------------------------------------------------------------------------------------**

**DELETE FROM бронь WHERE id\_бронь=**

**-------------------------------------------------------------------------------------------**

**select** *\** **from ОРГАНИЗАЦИЯ where ID\_ОРГАНИЗАЦИЯ=**

**-------------------------------------------------------------------------------------------**

**select** *\** **from БРОНЬ**

**-------------------------------------------------------------------------------------------**

**update ЖАЛОБА set СТАТУС\_ОБР="** + stat  
 + **" where ID\_ЖАЛОБА="** + id\_comp

**-------------------------------------------------------------------------------------------**

**insert into БРОНЬ values("** + (getMaxIdBooking() + 1) + **", "** + id + **", "** + data[7] + **", "**+ data[6] + **", "**+ data[1] + **", "**+ data[2] + **", "** + data[3] + **", date'"** + start.toString() + **"', date'"**+ finish.toString() + **"')"**;  
statement.executeQuery(sql)

**-------------------------------------------------------------------------------------------**

**insert into ПОСТОЯЛЕЦ values("** + idGuest + **", "** + idOrg + **", '"** + data[3] + **"')**

**-------------------------------------------------------------------------------------------**

**insert into ИСТОРИЯ values("** + (maxHistoryId() + 1)+ **", "** + idGuest + **", "** + idCorpus + **", "** + data[6] + **", date'"** + start.toString() + **"', date'"** + finish.toString() + **"', "** + data[4] + **")**

**-------------------------------------------------------------------------------------------**

**update номер set ЗАНЯТ='yes' where ID\_НОМЕР="** + data[6] + **" and ID\_КОРПУС=**

**-------------------------------------------------------------------------------------------**

**select** *\** **from ИСТОРИЯ where ID\_ПОСТОЯЛЕЦ="** + idGuest  
 + **" ORDER BY дата\_засел"**+**" desc**

**-------------------------------------------------------------------------------------------**

**select** *\** **from история\_услуг where id\_запись=**

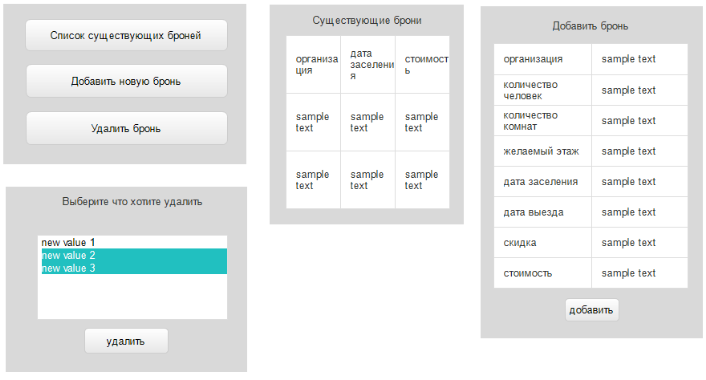
**-------------------------------------------------------------------------------------------**

**update история\_услуг set КОЛИЧЕСТВО="** + (count + 1)  
 + **"where id\_запись=**

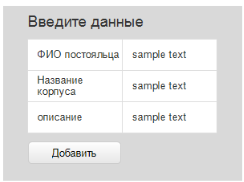
**-------------------------------------------------------------------------------------------**

**select ID\_ПОСТОЯЛЕЦ, ID\_НОМЕР from история where "** + **"ID\_КОРПУС="** + id\_cop  
 + **" and (date'"** + start+  
 **"'<= ДАТА\_ЗАСЕЛ and date'"** +  
 finish + **"' >= ДАТА\_ВЫЕЗД)"** + **" or (date'"** + start  
 + **"' between ДАТА\_ЗАСЕЛ and ДАТА\_ВЫЕЗД)"** + **" or (date'"** + finish  
 + **"' between ДАТА\_ЗАСЕЛ and ДАТА\_ВЫЕЗД)"** ;

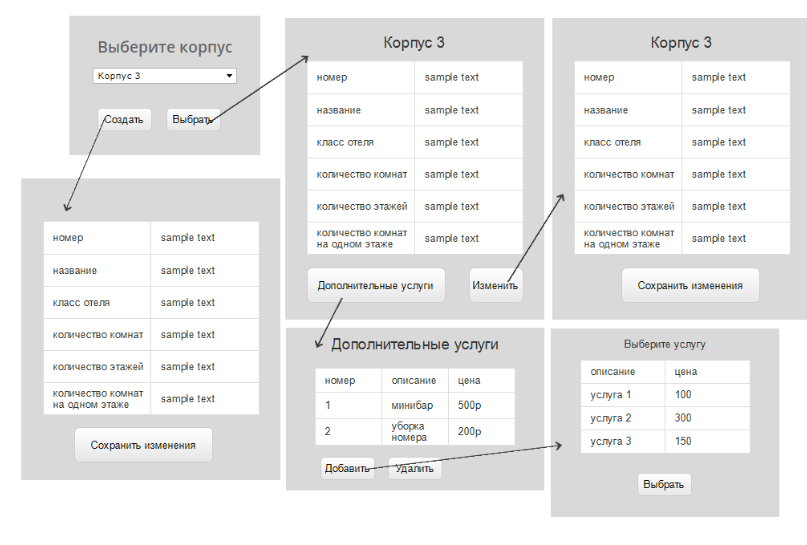
Форма «Бронь»



Форма «Добавить жалобу»



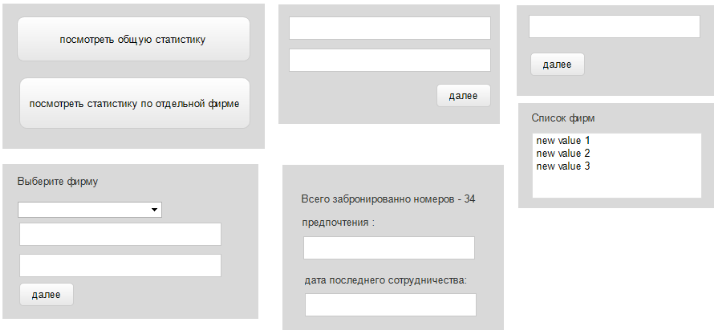
Форма «Информация о корпусе»



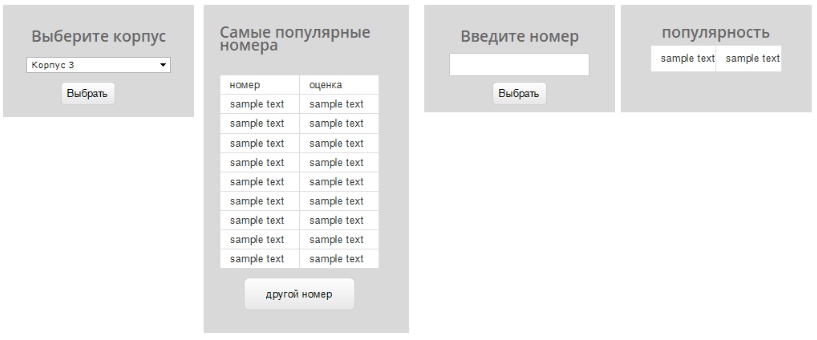
Форма «История заселений»



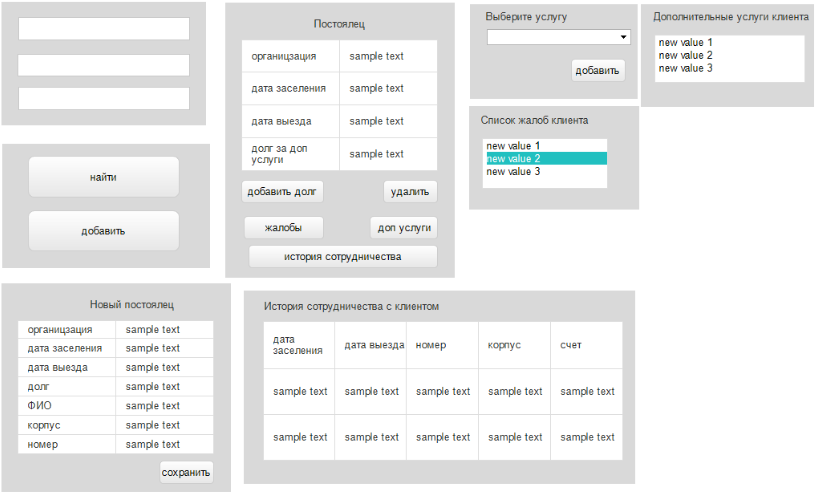
Форма «Организации»



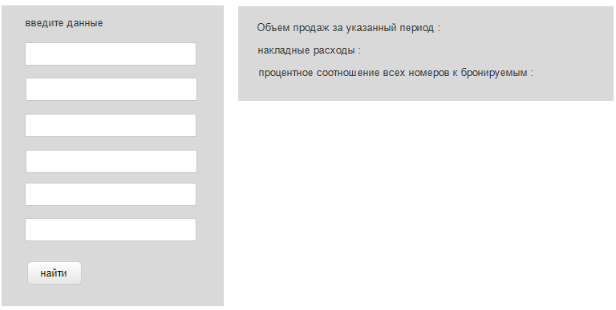
Форма «Популярные номера»



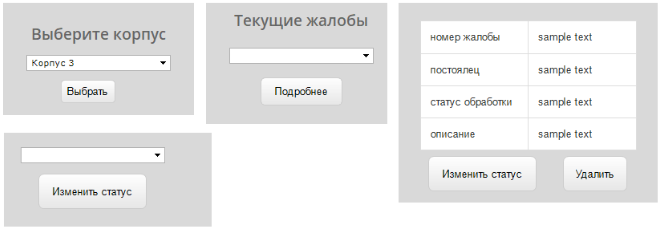
Форма «Постоялец»



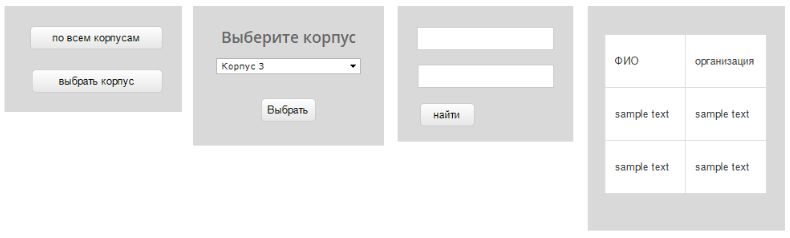
Форма «Статистика номеров»



Форма «Текущие жалобы»



Форма «Частые гости и новые клиенты»



Форма «Свободные и занятые номера»

