



Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка приложения «Отель» на Python»

Выполнил студент

Юшаков Никита Романович

Группа П1-17

_____ (Подпись)

_____ (Дата сдачи работы)

Проверил преподаватель

Гусятинер Леонид Борисович

_____ (Подпись)

_____ (Оценка)

Королёв 2020 г.

Оглавление

Введение	3
1. Теоретическая часть.....	4
1.1. Описание предметной области	4
1.2. Описание существующих разработок	9
1.2.1. Opera.	9
1.2.2. Fidelio.....	11
2. Проектная часть	13
2.1. Диаграмма прецедентов	13
2.2. Выбор инструментов.....	15
2.3. Проектирование сценария.....	16
2.4. Диаграмма классов	17
2.5. Описание главного модуля	19
2.6. Описание спецификаций к модулям	23
2.7. Описание модулей.....	25
2.8. Описание тестовых наборов модулей	27
2.9. Описание применения средств отладки	30
2.10. Анализ оптимальности использования памяти и быстродействия.....	31
3. Эксплуатационная часть	32
3.1. Руководство оператора.....	32
Заключение	40
Список литературы и интернет-источников.....	41
Приложение 1. Код главного модуля Implementation.py.	42
Приложение 2. Код модуля интерфейса UI.py.	44

Введение

Целью данного курсового проекта является написание программы «Отель» для упрощения работы отелей/гостиниц. Эта тема является актуальной на данный момент, потому что в наше время все стремятся все автоматизировать или упростить. Данный курсовой проект позволит облегчить работу персонала отеля, а именно администратора на ресепшене. Так же в данном проекте будет простой для понимания интерфейс и небольшой порог вхождения.

В первой части будет рассмотрена предметная область данной темы, а также несколько продуктов по данной теме.

Во второй части будут рассмотрены инструменты и модули, которые были разработаны, структура программной части и листинги ключевых частей программных модулей.

В третьей части будет рассмотрено руководство для пользователей.

В заключительной части будет приведен To-do лист с планами по доработки программы, а также сделаны общие выводы о получившемся проекте.

1. Теоретическая часть

1.1. Описание предметной области

Гостиничное дело – распространенная и прибыльная сфера бизнеса. Владелец может заниматься управлением самостоятельно, но организовать работу в этой сфере так, чтобы заведение приносило хорошую прибыль, сможет далеко не каждый.

Независимо от того, кто управляет бизнесом, в большинстве гостиниц и отелей предпочитают пользоваться специальной программой. Она позволяет грамотно распланировать совершение всех операций в заведении и обеспечивает удобное хранение информации о клиентах. Это открывает новые перспективы для развития отеля.

Современный отель представляет собой сложный хозяйственный комплекс, в управлении которым особо важна точность, оперативность и удобство, ведь номерной фонд гостиницы или отеля очень динамичен, постоянно происходят новые бронирования, отмены предыдущих броней, заселения и убытия постояльцев. И если не обеспечить максимальную эффективность учета этих процессов, то работа всей гостиницы будет нерезультативной.

Существует большое количество видов отелей: [4]

1. Бизнес-отель.

Бизнес-отели отличаются от других видов отелей, тем, что они нацелены на работу с определенной клиентурой – деловые люди. Именно поэтому, бизнес-отели, чаще всего располагаются в благоприятных для ведения бизнеса и переговоров местах: в центре города, в местах с прекрасно налаженным транспортным сообщением между ними и вокзалами, аэропортами, морскими портами, различными выставками и галереями, а также местными районами деловой активности. Помимо удобства транспортной инфраструктуры, бизнес-отели учитывают и другие потребности своих основных клиентов: в каждом номере имеется весь необходимый набор оргтехники. В отелях такого рода, часто располагаются

офисы разных компаний экспресс-доставки, информативное табло с биржевыми котировками, отели предоставляют возможность аренды своего бизнес-зала для проведения совещаний деловыми постояльцами.

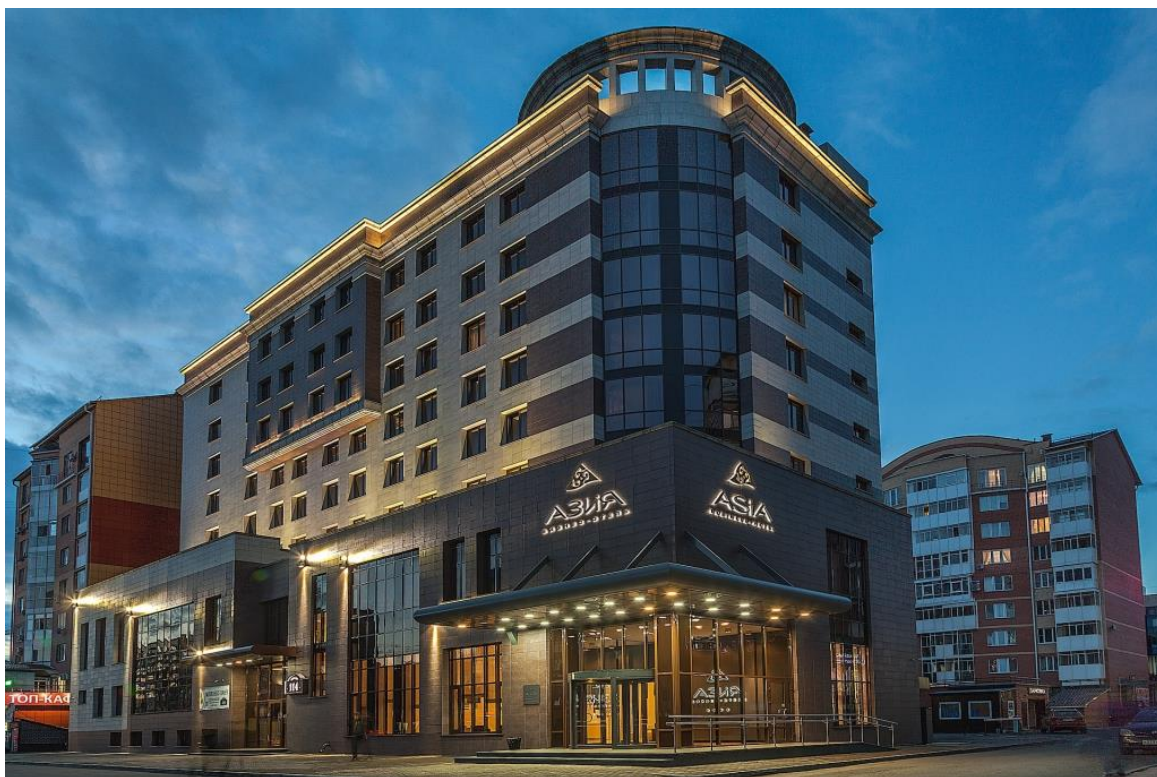


Рисунок 1. Бизнес отель Азия [5]

2. Отели-курорты или resort-отели.

В данных отелях имеется большой ассортимент инвентаря для занятий спортом и специализированные помещения, подходящие для этой цели, например, теннисный корт, бассейн, фитнес-центр. Отличается Resort-отель, от иного рода отелей своей закрытостью, наличием собственной службы безопасности. Resort-отель – автономная система организованного отдыха с большим выбором услуг для своих гостей. Это может быть наличие нескольких ресторанов национальных кухонь, свои магазины, парк развлечений или аквапарк, и это все на территории отеля. В отелях-Resorts – чаще всего имеется несколько зданий с разнообразными номерами, виллами с отдельными бассейнами, несколькими ресторанами и барами, иногда своими собственными линиями побережья.



Рисунок 2. Resort-отель [6]

3. Бутик-отели.

Бутик-отели – этот тип гостиницы с небольшим количеством номеров, но с уникальным дизайном и замечательным уровнем сервиса. Здание бутик-отеля должно иметь необычный или неповторимый дизайн всего интерьера, он должен иметь определенный стиль и все внутри и снаружи должно быть выдержано в рамках одной темы.

4. Апартаменты или апарт-отели.

Апартамент отеля сформировали именно семейные пары с детьми. Отличие апарт-отелей от других гостиниц в том, что гостям предлагаются не стандартные номера для проживания, а подходящие для обычной жизни квартиры. Номера апарт-отелей оснащены всем необходимым для повседневной жизни: техника на кухне, телевизор, компьютер, DVD, тренажерами.

5. Bed and Breakfast.

Отели Bed and Breakfast или B&B - это маленькие семейные отели, предлагающие своим клиентам домашний уют и комфорт, а также блюда

домашней кухни. Управление таким отелем осуществляет или владелец дома, или члены его семьи.

6. Хостелы.

В номере стоят несколько кроватей, на которых спят несколько человек различного пола. Не всегда на комнату приходится отдельный туалет и душ, порой он может быть один на весь этаж. Хостел – это большое общежитие с общей кухней, где стоит большой холодильник, там вы можете хранить свои продукты, в подписанной посуде. Порой хостелы предлагают своим клиентам открытый доступ в Интернет. Очень часто в хостелах имеется общая гостиная, где стоит телевизор.

7. СПА-отели.

Здесь можно рассчитывать на большой выбор оздоровительных процедур, различные лечебные курсы, разного рода массажи, посещение сауны, лечение целебными грязями, индивидуальные диеты, высоко квалифицированный уход за кожей, расслабляющие ванны с минеральной водой, сеансы ароматерапии, занятия йогой.

8. Мотели.

Мотель – некая разновидность отеля, являющаяся по своей сути малоэтажной придорожной гостиницей, с дверями номеров, выходящих на парковку, с очень аскетичной обстановкой. Весь смысл проживания в этих гостиницах заключен в том, чтобы провести тут ночь и двинуться дальше. Мотели можно встретить в любой стране мира. Цена проживания здесь невысок, но и уровень безопасности также невысок.

9. Пансионаты.

Пансионат - это вид отелей, где люди не только живут, отдыхают, но и проходят лечение. Чаще всего, пансионаты расположены в живописных природных местах, за городом, это может быть морское побережье или горная местность. Номерной фонд может быть и большим, но со скромной обстановкой и необходимой техникой: телевизором, радио, холодильником.



Рисунок 3. Пансионат [8]

10. Эко-отели.

Этот вид гостиниц стал в последнее десятилетие очень популярен. Их открывают везде. Главное условие эко-отеля – стопроцентно чистая вода, экологически чистые продукты питания и натуральные материалы, используемые в отделке номеров. Кроме того, эко-отели, чаще всего строят в заповедниках, где человек может гармонично соседствовать с природой, при этом, не загрязняя ее продуктами своей жизнедеятельности.

1.2. Описание существующих разработок

В этом разделе рассмотрены некоторые уже имеющиеся программы для управления отелями, а также цели таких программ и требования.

Интерфейс таких программ для управления отелями чаще всего простой и понятный для обычных пользователей.

Цели использования системы управления отелем:

1. Автоматизация работы персонала.
2. Синхронизация с системами бронирования.
3. Ведение бухгалтерии.
4. Оформление документов для постояльцев.
5. Расчет различных статистических показателей: в какое время года больше гостей, какие номера чаще бронируют.

Основные требования:

1. Ведение отчетности в соответствии с законодательством страны, в которой расположена недвижимость.
2. Размещение постояльцев по номерам.
3. Оплата при поселении или выезде.

1.2.1. Opera.

Полное название - система управления отелем Opera Enterprise Solution [3].

Одна из самых популярных систем управления отелям, в интернете есть множество положительных отзывов о системе.

Достоинства, которые подтверждают отзывы:

1. Универсальна, то есть может применяться в различных видах заведений гостиничного бизнеса.
2. Настраивается в зависимости от размера и объемов активности ведения бизнеса.
3. Соответствует самым требовательным запросам.

4. Гарантирует достоверный и оперативный обмен данными о недвижимости в режиме реального времени.

5. Имеет богатый функционал.

Также пользователи выделяют возможность работы программы со множеством других полезных для бизнеса программ, что является большим плюсом для любого бизнеса.

Система управления отелем Opera может работать вместе с такими программами, как:

1. Тарификация звонков.
2. Автоматические мини бары.
3. Авторизация кредитных карт.
4. Бухгалтерские программы.
5. Система управления ресторанами.



Рисунок 4. Интерфейс Opera

Данная система состоит из модулей, которые при необходимости могут быть настроены и изменены в соответствии с потребностями управляющего отеля. От этого зависит и стоимость разработки. Она включает в себя:

1. Бронирование.

Система управления бронированием отеля включает в себя подтверждение, аннуляцию, листы ожидания, управление депозитами.

2. Управление номерным фондом.

Автоматическое распределение заданий для горничных. Контроль наличия свободных номеров, их оснащения, уборки и своевременности ремонта.

3. Карта гостя.

Личные данные каждого постояльца: адрес, номер телефона, предпочтения по проживанию в гостинице, посещение ресторана, бассейна и другой инфраструктуры отеля.

4. Отчеты.

Возможность индивидуальной настройки с помощью встроенного генератора. Также можно воспользоваться стандартными отчетами (более 300 форм).

5. Управление тарифами.

Установка квот, анализ и управление доходами, настройка тарифов и многое другое.

6. Служба приема, размещения.

Размещение гостей по одному и группами, с бронированием и без него. Ведение служебных заметок и управление гостевыми сообщениями.

Система управления отелем Opera PMS затрагивает большинство деловых процессов заведения, осуществляет взаимодействие с различными интерфейсами: телефоны, телевидение, электронные замки, планшеты.

1.2.2. Fidelio

Система управления отелем Fidelio [3] – продукт немецкого производства.

В России Fidelio представляет компания Hotel and Restaurant systems (HRS). Она занимается реализацией и настройкой продуктов данной фирмы, помогает клиентам в освоении системы, оказывает техническую поддержку и проводит периодическое обновление.

Система управления гостиницей Fidelio не требует особых аппаратных характеристик компьютера и тоже очень популярна.

Особенности:

1. Цена зависит от количества номеров и от набора модулей, который вы выберете.
2. В стоимость входит установка, настройка, обучение клиентов, поддержка при неполадках.
3. Не всегда хорошо взаимодействует со струйными и лазерными принтерами.

Fidelio производит:

1. Начисление средств за проживание.
2. Бронирование и заселение.
3. Выставление счетов для клиентов.
4. Получение статистических отчетов.
5. Объединение данных о неоплаченных счетах постояльцев с различных точек продаж.

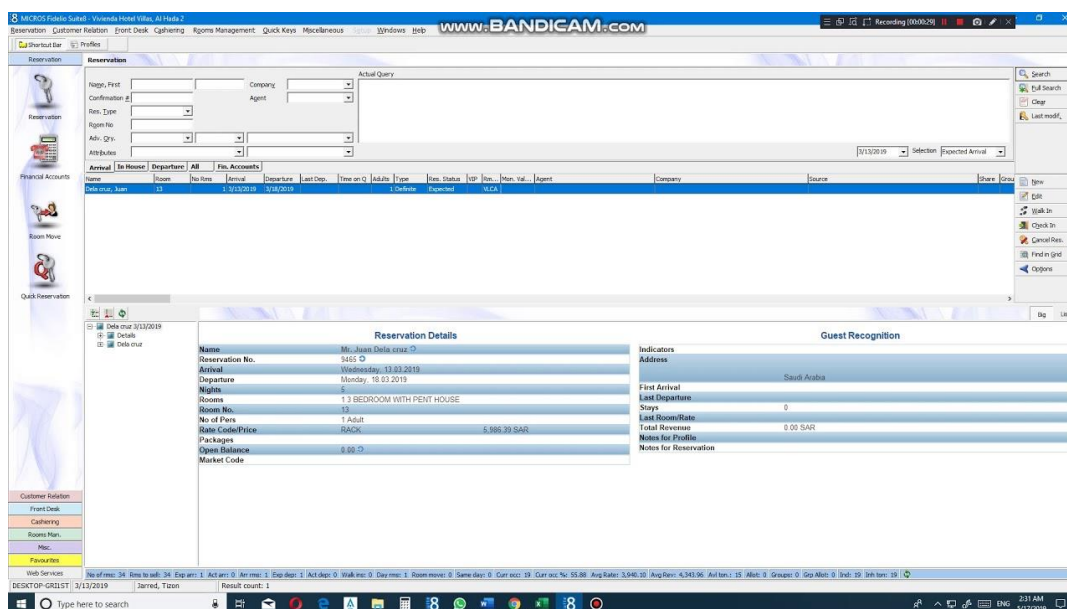


Рисунок 5. Окно бронирования номера

Судя по отзывам, большим плюсом этой системы является возможность взаимодействия с другими внешними устройствами и системами, что может быть очень удобно для любого бизнеса.

2. Проектная часть

2.1. Диаграмма прецедентов

В этом разделе представлены две диаграммы прецедентов. На первой диаграмме показаны все возможные функциональные и поведенческие отношения. На второй диаграмме показаны все возможные действия при бронировании номера гостем.

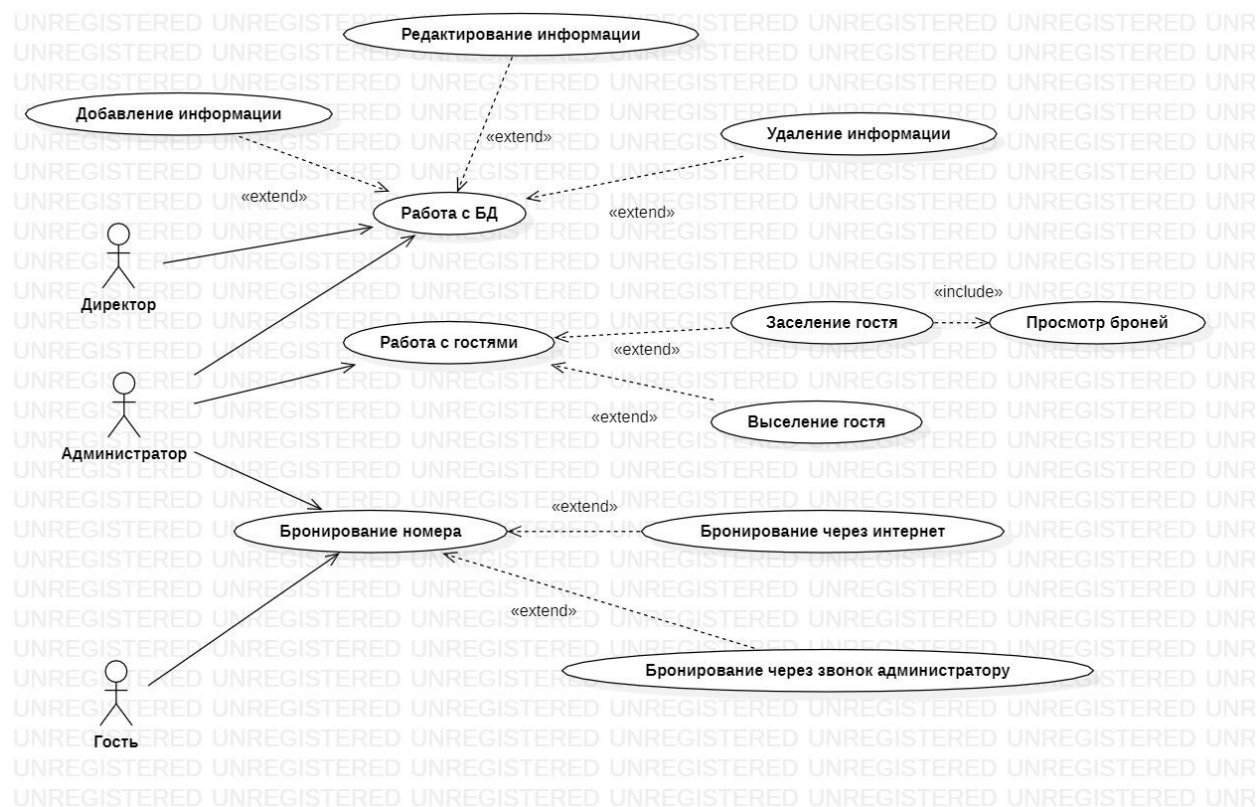


Рисунок 6. Диаграмма прецедентов работы на ресепшн

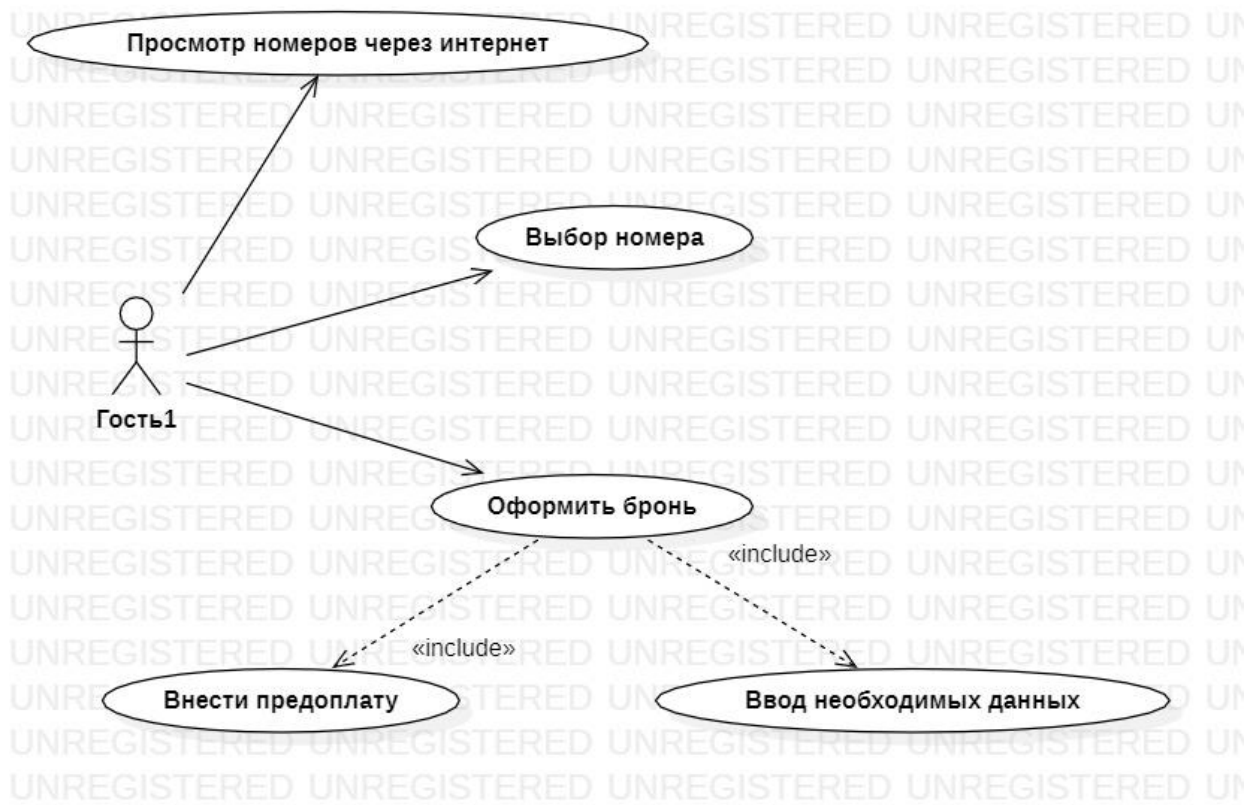


Рисунок 7. Диаграмма прецедентов бронирования через интернет

2.2. Выбор инструментов

При выборе инструментов было проведено сравнение по критериям, представленных в таблице 1.

Степень важности критерия выбиралась из: низкая, ниже средней, средняя, ниже высокой, высокая.

Таблица 1. Критерии выбора инструмента.

Критерий	Участие в корпоративном проекте	Простота сопровождения	Наличие библиотек	Наличие документации на русском языке	Скорость разработки
Важность критерия	Высокая	Средняя	Высокая	Ниже средней	Ниже высокой

Исходя из этих критериев, я сравнил 3 языка программирования от 0 до 10 баллов за критерий.

Таблица 2. Оценка языков программирования.

Критерий/Язык программирования	C++	Python	Object Pascal
Участие в корпоративном проекте	10	8	4
Простота сопровождения	7	10	3
Наличие библиотек	6	10	4
Наличие документации на русском языке	8	6	6
Скорость разработки	6	10	3
Итого баллов	37	44	20

По результатам сравнения был выбран язык программирования Python.

2.3. Проектирование сценария

В данном разделе приведен сценарий использования программы администратором отеля на ресепшине.

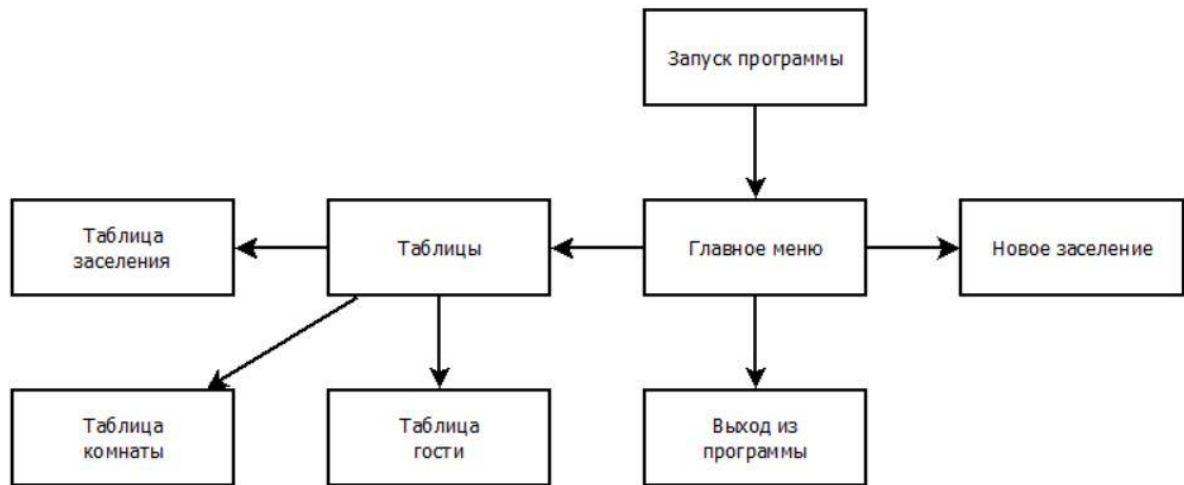


Рисунок 8. Сценарий использования

Пользователь после запуска программы может выполнить 3 действия: начать новое заселение, выйти из программы или начать работу с таблицами.

При выборе выхода программа заканчивает свою работу, при выборе работы с таблицами пользователь попадает в новое окно где может выбрать с какой таблицей работать и какие действия совершать.

2.4. Диаграмма классов

В данном разделе представлены все классы, используемые в проекте, а также их отношения между собой.

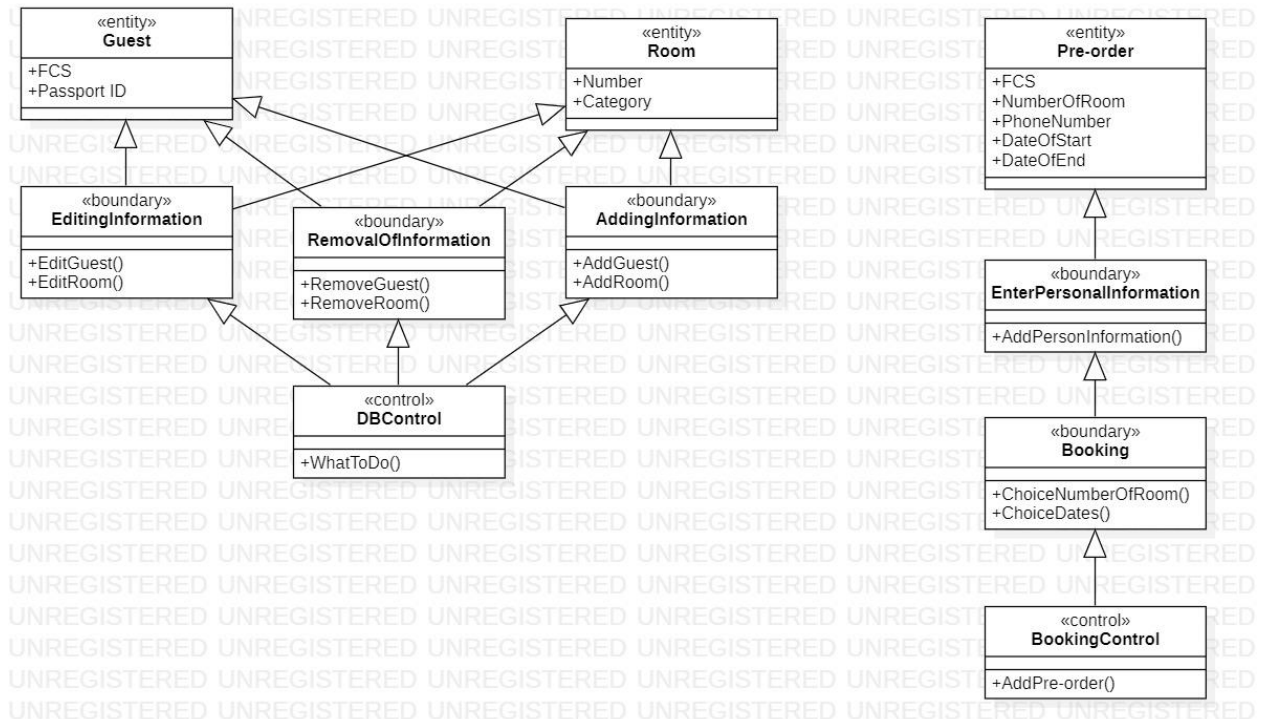


Рисунок 9. Диаграмма классов для проекта

Класс «Guest» содержит публичные поля FCS (ФИО) и Passport ID

Класс «Room» содержит публичные поля Number и Category

Класс «EditingInformation» является наследником классов «Guest» и «Room» и содержит публичные методы EditGuest(), который позволяет редактировать информацию о госте, а также EditRoom(), который позволяет редактировать информацию о комнате.

Класс «RemovalOfInformation» является наследником классов «Guest» и «Room» и содержит публичные методы RemoveGuest(), который позволяет удалять информацию о госте, а также RemoveRoom(), который позволяет удалять информацию о комнате.

Класс «AddingInformation» является наследником классов «Guest» и «Room» и содержит публичные методы AddGuest(), который позволяет добавлять информацию о госте, а также AddRoom(), который позволяет добавлять информацию о комнате.

Класс «DBControl» является наследником классов «EditingInformation», «RemovalOfInformation» и «AddingInformation» и содержит публичный метод WhatToDo(), который контролирует работу всех остальных методов классов-предков.

Класс «Pre-order» содержит публичные поля FCS, NumberOfRoom, PhoneNumber, DateOfStart, DateOfEnd.

Класс «EnterPersonalInformation» является наследником класса «Pre-order» и содержит публичный метод AddPersonInformation() для ввода гостем своих персональных данных

Класс «Booking» является наследником класса «EnterPersonalInformation» и содержит публичные методы ChoiceNumberOfRoom() для выбора номера и ChoiceDates() для выбора даты заезда и выезда из отеля

Класс «BookingControl» является наследником класса «Booking» и содержит публичный метод AddPre-order(), позволяющий гостю забронировать номер.

2.5. Описание главного модуля

В главный модуль входит класс `UiImplementation` который отвечает за функционал программы.

В главный модуль также импортируется еще 4 модуля, один из которых описывает интерфейс программы, один для работы с ОС, а два других входят в библиотеку `PyQt5` и позволяют работать с виджетами и основой интерфейса.

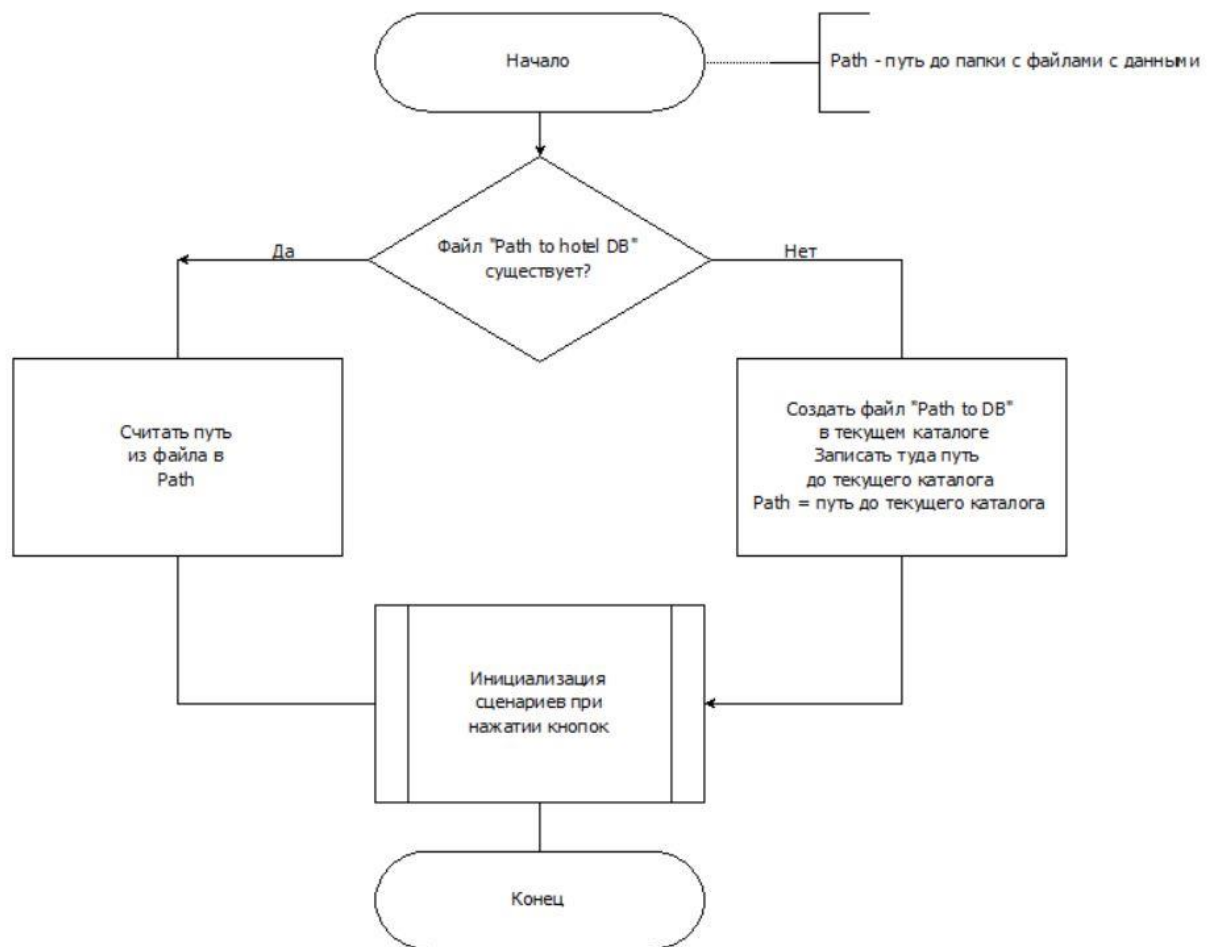
Также в главный модуль входит код, который запускает саму программу при ее запуске. Полный код главного модуля находится в приложении 1.

В классе `UiImplementation` реализованы следующие методы:

Листинг 1. Обработка всех событий в программе.

```
def __init__(self, interface):
    """Обработка всех событий в программе"""
    import os # Импорт модуля для работы с ОС
    if os.path.isfile(os.path.abspath(os.curdir) + "\\Path to hotel DB.txt"):
        with open(os.path.abspath(os.curdir) + "\\Path to hotel DB.txt") as
fp:
            self.path = fp.readline()
    else:
        with open(os.path.abspath(os.curdir) + "\\Path to hotel DB.txt", "w")
as fp:
            fp.write(os.path.abspath(os.curdir))
            self.path = os.path.abspath(os.curdir)
    self.setup_ui(interface)
    self.btn_exit.clicked.connect(QtWidgets.QApplication.exit)
    self.btn_show_the_tables.clicked.connect(self.open_table)
    self.btn_show_the_tables.clicked.connect(self.load_table)
    self.tables.currentIndexChanged.connect(self.load_table)
    self.btn_back.clicked.connect(self.back)
    self.btn_save.clicked.connect(self.save_table)
    self.btn_append.clicked.connect(self.append)
    self.btn_delete.clicked.connect(self.delete)
```

Данный метод отслеживает все действия пользователя и в зависимости от конкретного действия вызывает нужный метод. Блок-схема данного метода показана на рисунке ниже.

Рисунок 10. Блок-схема метода `__init__`**Листинг 2. Загрузка таблицы.**

```

def load_table(self):
    """Загрузка и вывод таблицы на экран"""
    import csv # Импорт модуля для работы с файлами в формате *.csv
    self.table.setSortingEnabled(False)
    self.table.setRowCount(0)
    self.table.setColumnCount(0)
    with open(self.path + "\\Hotel data bases\\" +
self.tables.itemText(self.tables.currentIndex()) + ".csv") as f:
        reader = csv.reader(f)
        for row_number, row_data in enumerate(reader):
            if row_number:
                self.table.setRowCount(row_number)
                for column_number, column_data in enumerate(row_data):
                    self.table.setItem(row_number - 1, column_number,
QtWidgets.QTableWidgetItem(column_data))
            else:
                self.table.setColumnCount(len(row_data))
                self.table.setHorizontalHeaderLabels(row_data)
        self.table.setSortingEnabled(True)
        self.table.resizeColumnsToContents()
  
```

Данный метод загружает и выводит на экран таблицу в зависимости от того какой пункт выбрал пользователь.

Листинг 3. Сохранение активной таблицы.

```
def save_table(self):
    """Сохранение таблицы"""
    import csv # Импорт модуля для работы с файлами в формате *.csv
    with open(self.path + "\\Hotel data bases\\" +
self.tables.itemText(self.tables.currentIndex()) + ".csv", "w",
        newline="") as f:
        writer = csv.writer(f)
        row_count = self.table.rowCount()
        column_count = self.table.columnCount()
        header = [self.table.horizontalHeaderItem(column).text() for column
in range(column_count)]
        writer.writerow(header)
        for row in range(row_count):
            row_data = list()
            for column in range(column_count):
                item = self.table.item(row, column)
                if item and item.text():
                    row_data.append(item.text())
            if row_data:
                writer.writerow(row_data)
```

Данный метод сохраняет изменения в активной таблице при нажатии на кнопку сохранить.

Листинг 4. Добавление поля в активную таблицу.

```
def append(self):
    """Добавление строки в таблицу"""
    self.table.setRowCount(self.table.rowCount() + 1)
```

Данный метод добавляет пустую строку в активную таблицу, что позволяет добавлять данные в таблицу с последующим сохранением этих данных.

Листинг 5. Удаление поля в активной таблице.

```
def delete(self):
    """Удаление выбранных строк из таблицы"""
    index_list = [QtCore.QPersistentModelIndex(model_index)
for model_index in
self.table.selectionModel().selectedRows()]
    for index in index_list:
        self.table.removeRow(index.row())
    self.save_table()
```

Данный метод удаляет выделенные строки в активной таблице.

Листинг 6. Переход к работе с таблицами.

```
def open_table(self):  
    """Переход из главного меню в режим работы с таблицами"""  
    Interface.setCurrentIndex(1)  
    self.tables.setCurrentIndex(0)
```

С помощью данного метода осуществляется переход из главного меню в окно работы с таблицами.

Листинг 7. Возвращение в главное меню.

```
def back(self):  
    """Возвращение в главное меню"""  
    Interface.setCurrentIndex(0)
```

С помощью данного метода осуществляется переход из любого окна в главное меню.

2.6. Описание спецификаций к модулям

В данном разделе описаны публичные члены модулей курсового проекта.

В главном модуле (Implementation.py) содержатся публичные методы класса UiImplementation который наследуется от класса UiInterface из модуля Ui.py. Так как все эти методы были описаны в разделе 2.5, в данном разделе описание данных методов опущено. В данном модуле также есть несколько объектов:

1. app – объект, который создает системное окно и объект самого приложения.
2. Interface – объект интерфейса представленный в виде стека окон.
3. ui – объект который представляет собой сам интерфейс.

В модуле, который реализует интерфейс (Ui.py) так же содержатся публичные методы класса UiInterface, которые будут описаны ниже в разделе 2.7, в данном разделе они описаны не будут. В данном классе так же есть публичные объекты, все они будут перечислены и описаны в этом разделе.

Объекты класса UiInterface:

1. interface – объект интерфейса представленный в виде стека окон.
2. verticalLayout_2 – вертикальный менеджер компоновки, позволяющий размещать и масштабировать объекты в зависимости от размера окна.
3. gridLayout – компоновщик с сеткой, который располагает объекты, находящиеся в нем в двумерную сетку.
4. spacerItem – разделитель, который позволяет прижать виджет (элемент интерфейса) к определенному углу или выравнить его по центру. Этот спейсер прижимает кнопки в главном меню к правому краю
5. spacerItem1 – разделитель, который позволяет прижать виджет (элемент интерфейса) к определенному углу или выравнить его по центру. Этот спейсер прижимает кнопки в главном меню к левому краю, тем самым выравнивая кнопки по центру

6. `verticalLayout` – вертикальный менеджер компоновки, позволяющий размещать и масштабировать объекты в зависимости от размера окна.
7. `gridLayout_3` - компоновщик с сеткой, который располагает объекты, находящиеся в нем в двумерную сетку.
8. `MainMenuPage` – страница главного меню
9. `name` – метка с названием программы
10. `btn_exit` – кнопка выхода
11. `btn_show_the_tables` – кнопка перехода в окно для работы с таблицами
12. `btn_new_check_in` – кнопка перехода в окно для нового заселения
13. `made_by` – метка с номером версии и разработчиком
14. `TablesPage` – страница для работы с таблицами
15. `tables` – combo box для выбора таблицы
16. `table` – виджет таблицы
17. `btn_save` – кнопка сохранения таблицы
18. `btn_append` – кнопка добавления строки в таблицу
19. `btn_delete` – кнопка удаления строк в таблице
20. `btn_back` – кнопка возврата в главное меню

2.7. Описание модулей

Кроме главного модуля программа содержит еще один модуль который реализует интерфейс (UI.py).

Полный код модуля приведен в приложении 2. В этом разделе приведены все методы, которые содержатся в класс UiInterface из модуля UI.py.

Листинг 8. Методы класса UiInterface.

```
def setup_ui(self, interface):
    """Создает интерфейс"""

def setup_main_menu_page(self):
    """Создает страницу главного меню"""

def setup_name(self):
    """Создает метку 'название программы'"""

def setup_btn_exit(self):
    """Создает кнопку выхода из программы"""

def setup_btn_show_the_tables(self):
    """Создает кнопку 'Tables'"""

def setup_btn_new_check_in(self):
    """Создает кнопку 'New check in' - в разработке"""

def setup_made_by(self):
    """Создает метку о создателе"""

def setup_tables_page(self):
    """Создает страницу с таблицами"""

def setup_tables(self):
    """Создает ComboBox для выбора таблицы"""

def setup_table(self):
    """Создает поле для отображения таблиц"""

def setup_btn_save(self):
    """Создает кнопку 'Save'"""

def setup_btn_append(self):
    """Создает кнопку 'Append'"""

def setup_btn_delete(self):
    """Создает кнопку 'Delete'"""

def setup_btn_back(self):
    """Создает кнопку 'Back'"""

def fill_in_ui(self, interface):
    """Заполняет все элементы нужными подписями"""
```



Рисунок 11. Блок-схема метода `setup_main_menu_page`

2.8. Описание тестовых наборов модулей

В этом разделе будут продемонстрированы результаты тестирования «черного ящика».

Тест 1. Смена таблицы.

Действия: Нажать на поле находящееся выше таблицы. Выбрать новую таблицу, в данном случае таблицу Rooms.

Ожидаемый результат: Смена таблицы «Check in» на таблицу «Rooms».

Результат теста:

Check in									
Guests									
Rooms									
2	Vadim Andreev Andreyevich	11	11	111112	12	23.05.2020	13.07.2020		
Append					Save				
Delete					Back				

Рисунок 12. Выбор таблицы

Rooms				
	Number	Category	Beds	Price/night
1	11	normal	1	200
2	12	low-cost	1	100
Append			Save	
Delete			Back	

Рисунок 13. Новая таблица

Тест 2. Добавление позиции в таблицу.

Действия: Нажать на кнопку «Append», в появившейся новой строке таблицы ввести необходимую информацию и нажать на кнопку «Save».

Ожидаемы результат: Добавление строки в таблицу, добавление в нее информации и сохранение отредактированной таблицы.

Результат тесте:

Rooms				
	Number	Category	Beds	Price/night
1	11	normal	1	200
2	12	low-cost	1	100
Append			Save	
Delete			Back	

Рисунок 14. Таблица комнат

Rooms				
	Number	Category	Beds	Price/night
1	11	normal	1	200
2	12	low-cost	1	100
3	13	normal	2	400
Append			Save	
Delete			Back	

Рисунок 15. Таблица комнат после добавления позиции

Тест 3. Удаление позиции из таблицы.

Действия: Нажать на цифру 3 в таблице, после выделения строки нажать кнопку «Delete».

Ожидаемый результат: Удаление 3 строки в таблице.

Результат теста:

Rooms				
	Number	Category	Beds	Price/night
1	11	normal	1	200
2	12	low-cost	1	100
3	13	normal	2	400

Append

Save

Delete

Back

Рисунок 16. Таблица комнат до удаления позиции

Rooms				
	Number	Category	Beds	Price/night
1	11	normal	1	200
2	12	low-cost	1	100

Append

Save

Delete

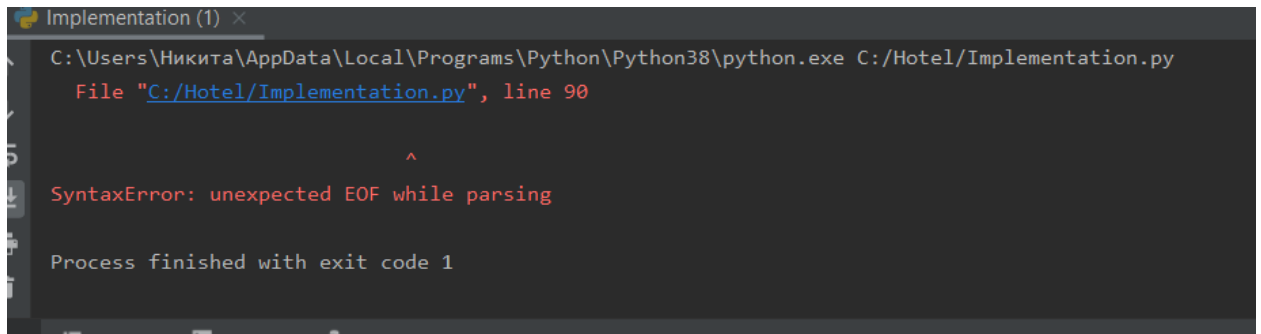
Back

Рисунок 17. Таблица комнат после удаления позиции

2.9. Описание применения средств отладки

В этом разделе показано умение применять средства отладки.

В ходе написания курсового проекта при попытке запустить скрипт было получено данное сообщение:

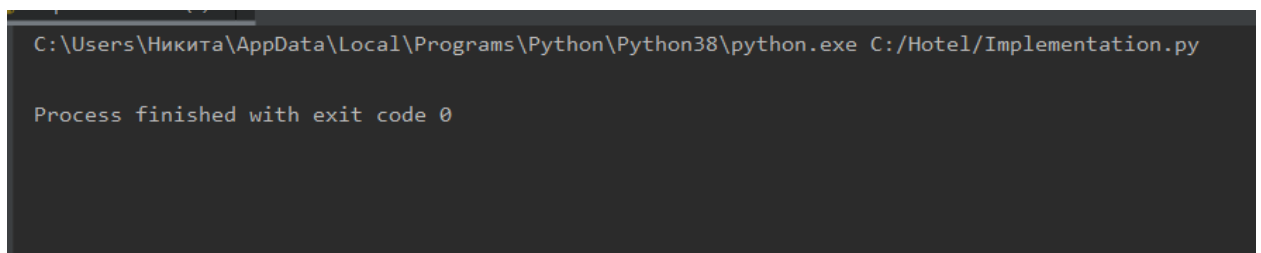


```
Implementation (1) x
C:\Users\Никита\AppData\Local\Programs\Python\Python38\python.exe C:/Hotel/Implementation.py
File "C:/Hotel/Implementation.py", line 90
    ^
SyntaxError: unexpected EOF while parsing

Process finished with exit code 1
```

Рисунок 18. До применения средств отладки

После получения данного сообщения были просмотрены 90 и 89 строки модуля Implementation.py и была обнаружена ошибка, которая впоследствии была устранена, а после попытки запуска скрипта получено данное сообщение:



```
C:\Users\Никита\AppData\Local\Programs\Python\Python38\python.exe C:/Hotel/Implementation.py

Process finished with exit code 0
```

Рисунок 19. После применения средств отладки

Это означает что ошибка была устранена и скрипт запустился.

2.10. Анализ оптимальности использования памяти и быстродействия

В данном разделе будет проведен анализ оптимальности использования памяти и быстродействия программы.

Список принятых оптимальных решений:

1. Подключение некоторых модулей внутри функций/методов.

В данном проекте некоторые модули были подключены не в весь модуль, а только в функции/методы, которые его используют. Сделано это потому что работа с локальными объектами быстрее работы с глобальными объектами, к тому же импортироваться эти модули будут только при срабатывании этих функций что ускорит запуск программы. Пример такого импортирования виден в разделе 2.5 в Листинг 1 или в Листинг 2.

2. Использование генераторов списков.

Использование генераторов списков не только упрощает чтение кода программистом из-за сокращения до 1 строки, но и позволяет ускорить данную операцию почти в 2 раза. Например, в разделе 2.5 в Листинг 5 используется генератор списка:

Листинг 9. Генератор списка заголовков

```
index_list = [QtCore.QPersistentModelIndex(model_index) for model_index
in self.table.selectionModel().selectedRows()]
```

Но можно было бы написать цикл:

Листинг 10. Цикл для получения заголовков

```
index_list = []
for model_index in self.table.selectionModel().selectedRows():
    index = QtCore.QPersistentModelIndex(model_index)
    index_list.append(index)
```

3. Эксплуатационная часть

3.1.Руководство оператора

АННОТАЦИЯ

В данном программном документе приведено руководство оператора по применению и эксплуатации программы «Hotel helper», предназначенной для облегчения работы отелей.

В данном программном документе, в разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы (минимальный состав аппаратных и программных средств и т.п.).

В данном программном документе, в разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77 ¹⁾, ГОСТ 19.103-77 ²⁾, ГОСТ 19.104-78* ³⁾, ГОСТ 19.105-78* ⁴⁾, ГОСТ 19.106-78* ⁵⁾, ГОСТ 19.505-79* ⁶⁾, ГОСТ 19.604-78* ⁷⁾).

¹⁾ ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов

²⁾ ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов

³⁾ ГОСТ 19.104-78* ЕСПД. Основные надписи

⁴⁾ ГОСТ 19.105-78* ЕСПД. Общие требования к программным документам

⁵⁾ ГОСТ 19.106-78* ЕСПД. Общие требования к программным документам, выполненным печатным способом

⁶⁾ ГОСТ 19.505-79* ЕСПД. Руководство оператора. Требования к содержанию и оформлению

⁷⁾ ГОСТ 19.604-78* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом

1. Назначение программы

1.1. Функциональное назначение программы

Специальное программное обеспечение «Hotel helper» используется для управления заездами, выездами в отель, отслеживания номеров отеля и хранения информации о гостях.

1.2. Эксплуатационное назначение программы

Специальное программное обеспечение «Hotel helper» может эксплуатироваться на объектах любого масштаба в сфере гостиничного бизнеса для облегчения работы персонала.

1.3. Состав функций

1.3.1. Функция открытия окна таблиц.

Эта функция позволяет перейти из главного меню в окно работы с таблицами

1.3.2. Функция смены таблицы

Эта функция позволяет менять таблицу в зависимости от необходимости.

1.3.3. Функция добавления элемента в таблицу.

Эта функция позволяет добавлять нужную информацию в таблицу.

1.3.4. Функция удаления элемента из таблицы.

Эта функция позволяет удалить ненужные элементы в таблице.

1.3.5. Функция возвращения в главное меню.

Эта функция позволяет вернуться в главное меню.

2. Условия выполнения программы

2.1. Минимальный состав аппаратных средств

ОС: Windows 10

Процессор: Как минимум 1 ГГц или SoC.

ОЗУ: 1 ГБ (для 32-разрядных систем) или 2 ГБ (для 64-разрядных систем).

Место на жестком диске: 16 ГБ (для 32-разрядных систем) или 20 ГБ (для 64-разрядных систем).

Видеоадаптер: DirectX версии не ниже 9 с драйвером WDDM 1.0.

Дисплей: 800 x 600.

2.2. Минимальный состав программных средств

Дополнительные программные средства не требуются.

2.3. Требование к персоналу (пользователю)

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

3. Выполнение программы

3.1. Загрузка и запуск программы

Перед запуском программы вам необходимо создать каталог «Hotel data bases» а в этом каталоге 3 файла: «Check in.csv» «Guests.csv» «Rooms.csv». В этих файлах прописать через запятую поля, которые вы хотите видеть в таблицах в программе, как на рисунке ниже.

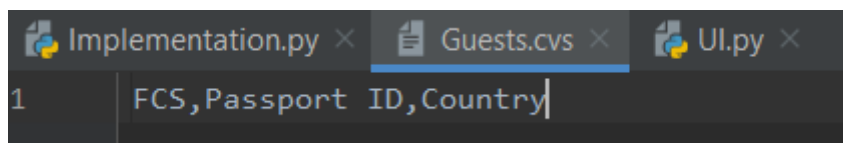


Рисунок 20. Поля таблицы Guests

Если каталог «Hotel data bases» находится не в каталоге с программой, вам нужно создать файл «Path to hotel DB.txt» в каталоге с программой и прописать туда путь до этого каталога. Если каталог «Hotel data bases» находится в каталоге с программой, файл «Path to hotel DB.txt» будет создан автоматически после первого запуска программы.

Важно! Путь до каталога «Hotel data bases» должен содержать только латиницу.

Запустите программу «Hotel helper», откроется окно главного меню:



Рисунок 21. Главное меню

Кнопки меню имеют следующие функции:

Tables – открывает окно для работы с таблицами.

Exit – Выход из программы.

3.2.Выполнение программы

3.2.1. Выполнение функции открытия окна таблиц.

Выберите пункт меню «Tables», после этого откроется окно с таблицами.

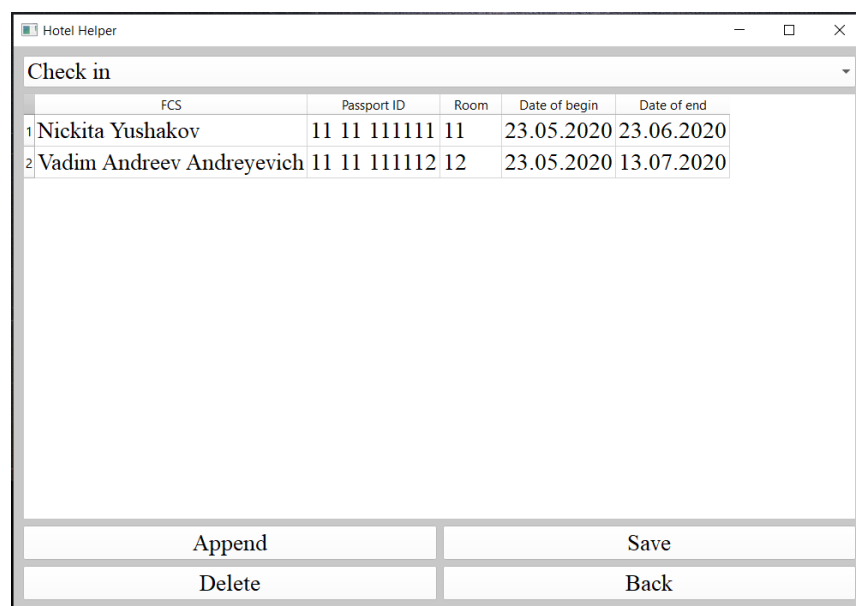


Рисунок 22. Окно для работы с таблицами

3.2.2. Выполнение функции смены таблицы.

Нажмите на поле находящееся выше таблицы. Выберите новую таблицу, в данном случае таблицу «Guests».

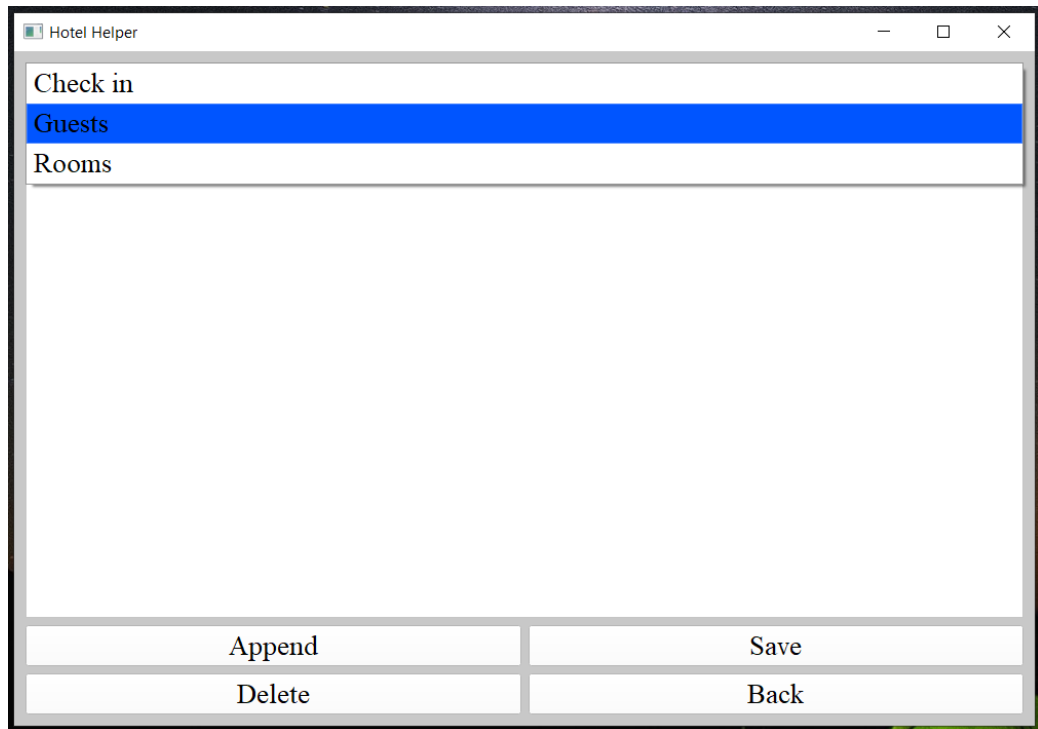


Рисунок 23. Выбор таблицы

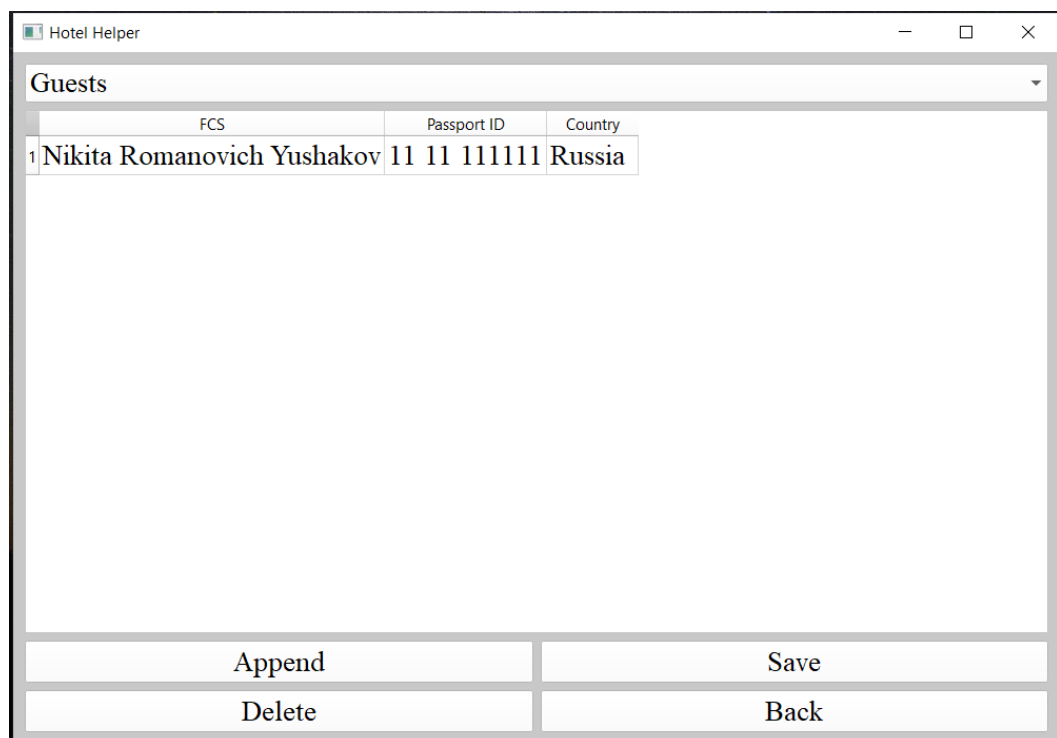


Рисунок 24. Новая таблица

3.2.3. Выполнение функции добавления элемента в таблицу.

Нажмите на кнопку «Append», появится новое поле в выбранной таблице, введите всю необходимую информацию в таблицу и нажмите на кнопку «Save», если этого не сделать, то все что вы ввели не будет сохранено.

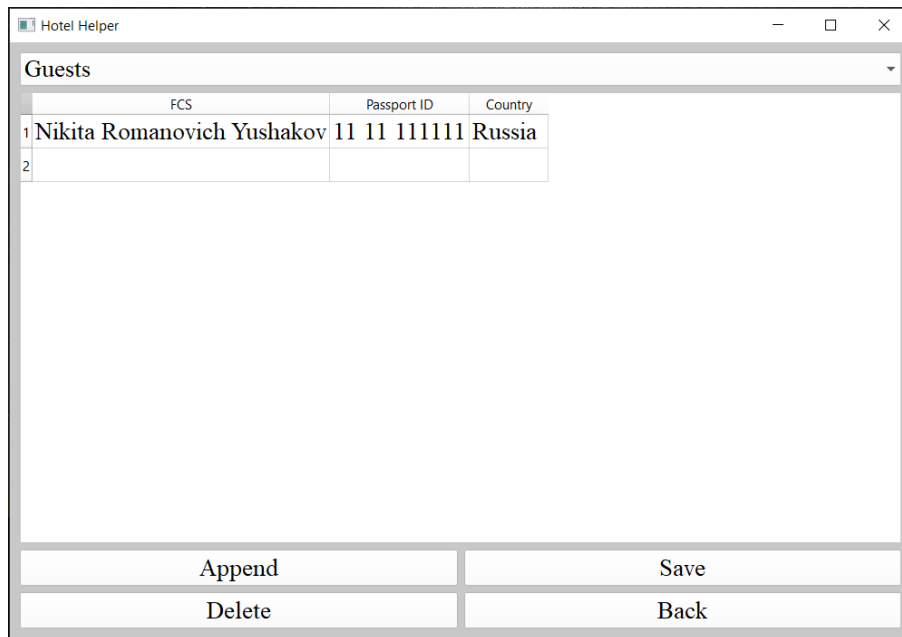


Рисунок 25. Таблица «Guests» после нажатия «Append»

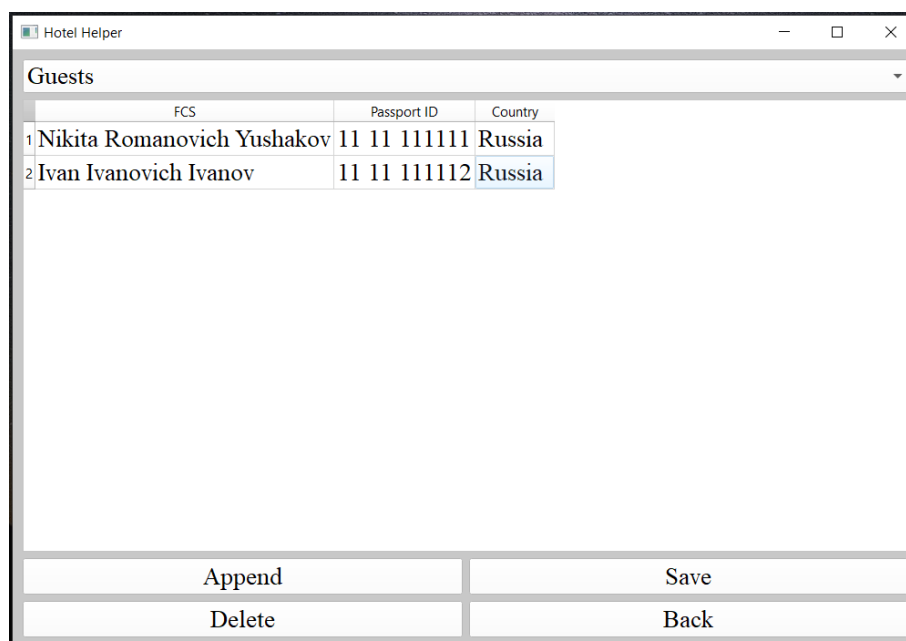


Рисунок 26. Таблица «Guests» после ввода информации и нажатия на кнопку «Save»

3.2.4. Выполнение функции удаления элемента из таблицы.

Нажмите на номер строки, которую вы хотите удалить и нажмите на кнопку «Delete». Будьте внимательны, удаленные данные нельзя восстановить!

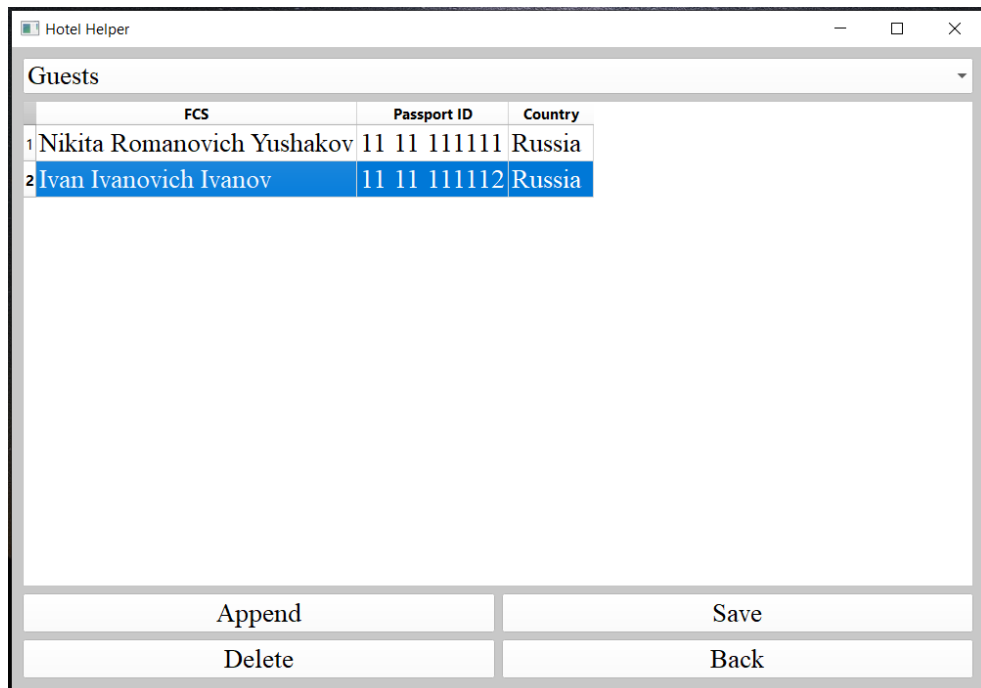


Рисунок 27. Элемент для удаления

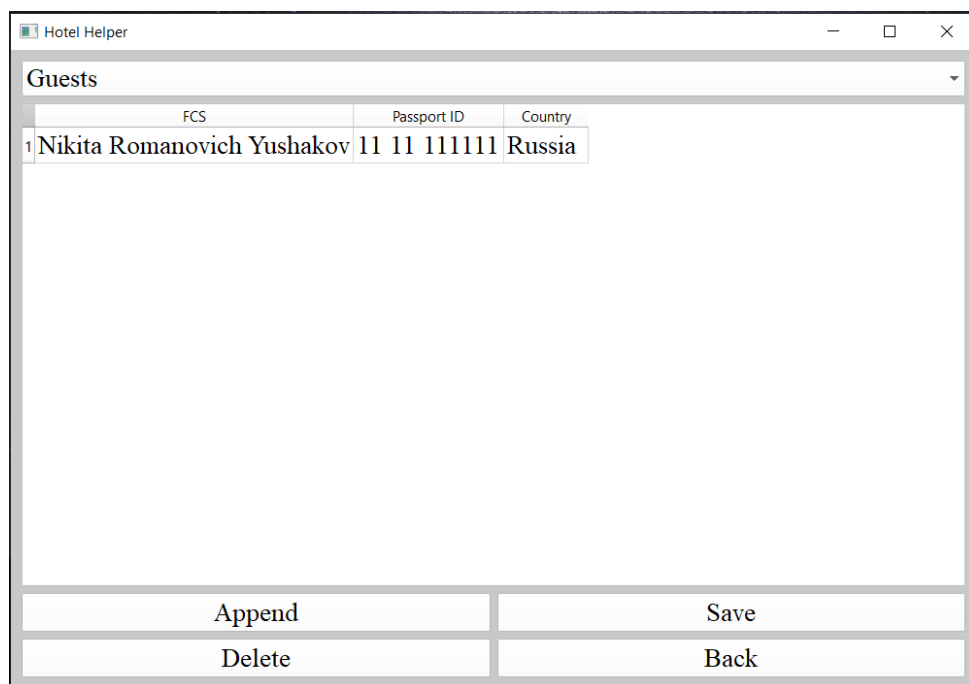


Рисунок 28. Таблица после удаления элемента

3.2.5. Выполнение функции возвращения в главное меню.

Чтобы вернуться в главное меню нажмите кнопку «Back».

3.3. Завершение работы программы

Чтобы завершить работу программы нажмите в главном меню на кнопку «Exit» или на кнопку «Закреть» в верхнем правом углу окна.

Заключение

В результате выполнения курсового проекта была написана программа «Hotel helper» для упрощения ведения бизнеса в гостиничной сфере, а именно для отслеживания заездов, выездов, гостей и номеров в отелях и гостиницах.

В ходе работы были проанализированы предметная область, существующие разработки, посвященные данному направлению, получены практические навыки по созданию UI с помощью библиотеки PyQt5.

Также планируется продолжать работу над данным проектом с целью расширения возможностей и удобства приложения для пользователей. Планы по доработкам представлены ниже.

To-do лист:

1. Доработка всех пунктов главного меню, а именно функции нового заселения в виде отдельной опции для удобства работы.
2. Добавления расчета цены за проживание
3. Доработка интерфейса с целью упрощения работы с программой.
4. Перенос всех данных в БД.
5. Улучшение дизайна интерфейса.

Список литературы и интернет-источников

1. Гагарина, Л. Г. Технология разработки программного обеспечения: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под ред. Л.Г. Гагариной. — Москва: ИД «ФОРУМ»: ИНФРА-М, 2019. — 400 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-104071-3. - Текст: электронный. - URL: <https://znanium.com/catalog/product/1011120>
2. Статья про системы управления отелями: <https://fb.ru/article/413610/sistema-upravleniya-otelem-obzor-luchshih-programm-vozmojnosti-opisanie-otzyivyi>
3. Сайт представителя Opera и Fidelio в России: <https://www.hrsinternational.com/russia/ru/>
4. Разновидность отелей: <https://puteshestvie.net/hotels/136-kakie-byvayut-vidy-oteley.html>
5. Бизнес-отель Азия: <https://pro-oteli.ru/hotels/126/6824/>
6. Resort отель Garden Cliff Resort & Spa: <https://market-sletat.ru/countries/thailand/hotels/pattajya/naklua/garden-cliff-resort-spa/>
7. Пансионат Фея-1: <https://kuban-kurort.com/objects/anapa/djemete/pansionat/209-feya-1/photos/>

Приложение 1. Код главного модуля Implementation.py.

```

"""Модуль реализующий работу программы"""
from UI import UiInterface # Импорт класса реализующего интерфейс
from PyQt5 import QtWidgets, QtCore # Импорт модулей для работы с
интерфейсом

class UiImplementation(UiInterface):
    """Класс реализаии работы интерфейса"""

    def __init__(self, interface):
        """Обработка всех событий в программе"""
        import os # Импорт модуля для работы с ОС
        if os.path.isfile(os.path.abspath(os.curdir) + "\\Path to hotel
DB.txt"):
            with open(os.path.abspath(os.curdir) + "\\Path to hotel DB.txt")
as fp:
                self.path = fp.readline()
            else:
                with open(os.path.abspath(os.curdir) + "\\Path to hotel DB.txt",
"w") as fp:
                    fp.write(os.path.abspath(os.curdir))
                    self.path = os.path.abspath(os.curdir)
        self.setup_ui(interface)
        self.btn_exit.clicked.connect(QtWidgets.QApplication.exit)
        self.btn_show_the_tables.clicked.connect(self.open_table)
        self.btn_show_the_tables.clicked.connect(self.load_table)
        self.tables.currentIndexChanged.connect(self.load_table)
        self.btn_back.clicked.connect(self.back)
        self.btn_save.clicked.connect(self.save_table)
        self.btn_append.clicked.connect(self.append)
        self.btn_delete.clicked.connect(self.delete)

    def load_table(self):
        """Загрузка и вывод таблицы на экран"""
        import csv # Импорт модуля для работы с файлами в формате *.csv
        self.table.setSortingEnabled(False)
        self.table.setRowCount(0)
        self.table.setColumnCount(0)
        with open(self.path + "\\Hotel data bases\\" +
self.tables.itemText(self.tables.currentIndex()) + ".csv") as f:
            reader = csv.reader(f)
            for row_number, row_data in enumerate(reader):
                if row_number:
                    self.table.setRowCount(row_number)
                    for column_number, column_data in enumerate(row_data):
                        self.table.setItem(row_number - 1, column_number,
QtWidgets.QTableWidgetItem(column_data))
                else:
                    self.table.setColumnCount(len(row_data))
                    self.table.setHorizontalHeaderLabels(row_data)
        self.table.setSortingEnabled(True)
        self.table.resizeColumnsToContents()

```

```

def save_table(self):
    """Сохранение таблицы"""
    import csv # Импорт модуля для работы с файлами в формате *.csv
    with open(self.path + "\\Hotel data bases\\" +
self.tables.itemText(self.tables.currentIndex()) + ".csv", "w",
        newline="") as f:
        writer = csv.writer(f)
        row_count = self.table.rowCount()
        column_count = self.table.columnCount()
        header = [self.table.horizontalHeaderItem(column).text() for
column in range(column_count)]
        writer.writerow(header)
        for row in range(row_count):
            row_data = list()
            for column in range(column_count):
                item = self.table.item(row, column)
                if item and item.text:
                    row_data.append(item.text())
            if row_data:
                writer.writerow(row_data)

def append(self):
    """Добавление строки в таблицу"""
    self.table.setRowCount(self.table.rowCount() + 1)

def delete(self):
    """Удаление выбранных строк из таблицы"""
    index_list = [QtCore.QPersistentModelIndex(model_index)
for model_index in
self.table.selectionModel().selectedRows()]
    for index in index_list:
        self.table.removeRow(index.row())
    self.save_table()

def open_table(self):
    """Переход из главного меню в режим работы с таблицами"""
    Interface.setCurrentIndex(1)
    self.tables.setCurrentIndex(0)

def back(self):
    """Возвращение в главное меню"""
    Interface.setCurrentIndex(0)

if __name__ == "__main__":
    import sys # Импорт модуля для работы с системой
    QtWidgets.QApplication.setStyle("Fusion")
    app = QtWidgets.QApplication(sys.argv)
    Interface = QtWidgets.QStackedWidget()
    ui = UiImplementation(Interface)
    Interface.show()
    sys.exit(app.exec_())

```

Приложение 2. Код модуля интерфейса UI.py.

```

"""Модуль реализующий интерфейс программы"""
from PyQt5 import QtCore, QtGui, QtWidgets # Импорт модулей для работы с
интерфейсом

class UiInterface(object):
    def setup_ui(self, interface):
        """Заполняет интерфейс"""
        interface.setObjectName("Interface")
        interface.resize(994, 657)

        self.setup_main_menu_page()

        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.MainMenuPage)
        self.verticalLayout_2.setObjectName("verticalLayout_2")

        self.setup_name()

        self.gridLayout = QtWidgets.QGridLayout()
        self.gridLayout.setVerticalSpacing(50)
        self.gridLayout.setObjectName("gridLayout")

        self.setup_btn_exit()

        spacerItem = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout.addItem(spacerItem, 2, 0, 1, 1)

        self.setup_btn_show_the_tables()

        spacerItem1 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.gridLayout.addItem(spacerItem1, 2, 2, 1, 1)

        self.setup_btn_new_check_in()

        self.setup_made_by()

        interface.addWidget(self.MainMenuPage)

        self.setup_tables_page()

        self.verticalLayout = QtWidgets.QVBoxLayout(self.TablesPage)
        self.verticalLayout.setObjectName("verticalLayout")

        self.setup_tables()

        self.setup_table()

        self.gridLayout_3 = QtWidgets.QGridLayout()
        self.gridLayout_3.setObjectName("gridLayout_3")

        self.setup_btn_save()

```



```

self.setup_btn_append()

self.setup_btn_delete()

self.setup_btn_back()

interface.addWidget(self.TablesPage)

self.fill_in_ui(interface)
interface.setCurrentIndex(0)
QtCore.QMetaObject.connectSlotsByName(interface)

def setup_main_menu_page(self):
    """Создает страницу главного меню"""
    self.MainMenuPage = QtWidgets.QWidget()
    self.MainMenuPage.setAutoFillBackground(False)
    self.MainMenuPage.setStyleSheet("background-color: rgb(200, 200,
200);")
    self.MainMenuPage.setObjectName("MainMenuPage")

def setup_name(self):
    """Создает метку 'название программы'"""
    self.name = QtWidgets.QLabel(self.MainMenuPage)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(72)
    font.setItalic(False)
    self.name.setFont(font)
    self.name.setAlignment(QtCore.Qt.AlignCenter)
    self.name.setObjectName("name")
    self.verticalLayout_2.addWidget(self.name)

def setup_btn_exit(self):
    """Создает кнопку выхода из программы"""
    self.btn_exit = QtWidgets.QPushButton(self.MainMenuPage)
    sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Fixed)
    sizePolicy.setHorizontalStretch(0)
    sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.btn_exit.sizePolicy().hasHeightForWidth())
    self.btn_exit.setSizePolicy(sizePolicy)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(16)
    self.btn_exit.setFont(font)
    self.btn_exit.setStyleSheet("background-color: rgb(255, 255, 255);")
    self.btn_exit.setObjectName("btn_exit")
    self.gridLayout.addWidget(self.btn_exit, 2, 1, 1, 1)

def setup_btn_show_the_tables(self):
    """Создает кнопку 'Tables'"""
    self.btn_show_the_tables = QtWidgets.QPushButton(self.MainMenuPage)
    sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,

```

```

QtWidgets.QSizePolicy.Fixed)
    sizePolicy.setHorizontalStretch(0)
    sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.btn_show_the_tables.sizePolicy().hasHeightF
orWidth())
    self.btn_show_the_tables.setSizePolicy(sizePolicy)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(16)
    self.btn_show_the_tables.setFont(font)
    self.btn_show_the_tables.setAcceptDrops(False)
    self.btn_show_the_tables.setStyleSheet("background-color: rgb(255,
255, 255);")
    self.btn_show_the_tables.setObjectName("btn_show_the_tables")
    self.gridLayout.addWidget(self.btn_show_the_tables, 1, 1, 1, 1)

def setup_btn_new_check_in(self):
    """Создает кнопку 'New check in' - в разработке"""
    self.btn_new_check_in = QtWidgets.QPushButton(self.MainMenuPage)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(16)
    self.btn_new_check_in.setFont(font)
    self.btn_new_check_in.setStyleSheet("background-color: rgb(255, 255,
255);")
    self.btn_new_check_in.setObjectName("btn_new_check_in")
    self.gridLayout.addWidget(self.btn_new_check_in, 0, 1, 1, 1)
    self.verticalLayout_2.addLayout(self.gridLayout)

def setup_made_by(self):
    """Создает метку о создателе"""
    self.made_by = QtWidgets.QLabel(self.MainMenuPage)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(10)
    self.made_by.setFont(font)
    self.made_by.setAlignment(QtCore.Qt.AlignBottom |
QtCore.Qt.AlignLeading | QtCore.Qt.AlignLeft)
    self.made_by.setObjectName("made_by")
    self.verticalLayout_2.addWidget(self.made_by)

def setup_tables_page(self):
    """Создает страницу с таблицами"""
    self.TablesPage = QtWidgets.QWidget()
    self.TablesPage.setStyleSheet("background-color: rgb(200, 200,
200);")
    self.TablesPage.setObjectName("TablesPage")

def setup_tables(self):
    """Создает ComboBox для выбора таблицы"""
    self.tables = QtWidgets.QComboBox(self.TablesPage)
    sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Fixed)
    sizePolicy.setHorizontalStretch(0)

```

```

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.tables.sizePolicy().hasHeightForWidth())
self.tables.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setFamily("Times New Roman")
font.setPointSize(16)
self.tables.setFont(font)
self.tables.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
self.tables.setTabletTracking(False)
self.tables.setLayoutDirection(QtCore.Qt.LeftToRight)
self.tables.setAutoFillBackground(False)
self.tables.setStyleSheet("background-color: rgb(255, 255, 255);\n"
                           "selection-background-color: rgb(0, 85,
255);\n"
                           "color: rgb(0, 0, 0);")
self.tables setFrame(False)
self.tables.setModelColumn(0)
self.tables.setObjectName("tables")
self.tables.addItem("")
self.tables.addItem("")
self.tables.addItem("")
self.verticalLayout.addWidget(self.tables)

def setup_table(self):
    """Создает поле для отображения таблиц"""
    self.table = QtWidgets.QTableWidget(self.TablesPage)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(16)
    self.table.setFont(font)

self.table.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustIgnored)
self.table.setWordWrap(True)
self.table.setCornerButtonEnabled(False)
self.table.setObjectName("table")
self.table.setColumnCount(0)
self.table.setRowCount(0)
self.table.horizontalHeader().setCascadingSectionResizes(True)
self.table.verticalHeader().setCascadingSectionResizes(True)
self.verticalLayout.addWidget(self.table)

def setup_btn_save(self):
    """Создает кнопку 'Save'"""
    self.btn_save = QtWidgets.QPushButton(self.TablesPage)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(16)
    self.btn_save.setFont(font)
    self.btn_save.setStyleSheet("background-color: rgb(255, 255, 255);")
    self.btn_save.setObjectName("btn_save")
    self.gridLayout_3.addWidget(self.btn_save, 0, 1, 1, 1)

def setup_btn_append(self):
    """Создает кнопку 'Append'"""

```

```

self.btn_append = QtWidgets.QPushButton(self.TablesPage)
font = QtGui.QFont()
font.setFamily("Times New Roman")
font.setPointSize(16)
self.btn_append.setFont(font)
self.btn_append.setStyleSheet("background-color: rgb(255, 255,
255);")

self.btn_append.setObjectName("btn_append")
self.gridLayout_3.addWidget(self.btn_append, 0, 0, 1, 1)

def setup_btn_delete(self):
    """Создает кнопку 'Delete'"""
    self.btn_delete = QtWidgets.QPushButton(self.TablesPage)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(16)
    self.btn_delete.setFont(font)
    self.btn_delete.setStyleSheet("background-color: rgb(255, 255,
255);")

    self.btn_delete.setObjectName("btn_delete")
    self.gridLayout_3.addWidget(self.btn_delete, 2, 0, 1, 1)

def setup_btn_back(self):
    """Создает кнопку 'Back'"""
    self.btn_back = QtWidgets.QPushButton(self.TablesPage)
    font = QtGui.QFont()
    font.setFamily("Times New Roman")
    font.setPointSize(16)
    self.btn_back.setFont(font)
    self.btn_back.setContextMenuPolicy(QtCore.Qt.PreventContextMenu)
    self.btn_back.setToolTipDuration(-1)
    self.btn_back.setStyleSheet("background-color: rgb(255, 255, 255);")
    self.btn_back.setObjectName("btn_back")
    self.gridLayout_3.addWidget(self.btn_back, 2, 1, 1, 1)
    self.verticalLayout.addLayout(self.gridLayout_3)

def fill_in_ui(self, interface):
    """Заполняет все элементы нужными подписями"""
    _translate = QtCore.QCoreApplication.translate
    interface.setWindowTitle(_translate("Interface", "Hotel Helper"))
    self.name.setText(_translate("Interface", "Hotel Helper"))
    self.btn_exit.setText(_translate("Interface", "Exit"))
    self.btn_show_the_tables.setText(_translate("Interface", "Tables"))
    self.btn_new_check_in.setText(_translate("Interface", "Under
construction"))
    self.made_by.setText(_translate("Interface", "Ver. 0.6 11.06.2020 by
Nikita Yushakov"))
    self.tables.setItemText(0, _translate("Interface", "Check in"))
    self.tables.setItemText(1, _translate("Interface", "Guests"))
    self.tables.setItemText(2, _translate("Interface", "Rooms"))
    self.table.setStyleSheet(_translate("Interface", "background-color:
rgb(255, 255, 255);"))
    self.table.setSortingEnabled(False)
    self.btn_save.setText(_translate("Interface", "Save"))
    self.btn_append.setText(_translate("Interface", "Append"))

```

```
self.btn_delete.setText(_translate("Interface", "Delete"))  
self.btn_back.setText(_translate("Interface", "Back"))
```