МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное

бюджетное образовательное учреждение

высшего образования

**«Московский технический университет связи и информатики»**

**(МТУСИ)**

Кафедра

«Математическая кибернетика и информационные технологии»

Проектная работа

По дисциплине «Программирование на языке Python»

Выполнили студенты:

М092301(75) Леонов Николай

М092301(75) Бахышов Фарид

Проверил:

ст. пр. Кузин А.А.

Москва 2023

Содержание

[ВВЕДЕНИЕ 3](#_Toc154421143)

[1. Анализ существующих решений 4](#_Toc154421144)

[2. Проектирование системы 7](#_Toc154421145)

[2.1. Определение требований к системе 7](#_Toc154421146)

[2.2. Структура системы 7](#_Toc154421147)

[3. Реализация разрабатываемой системы 10](#_Toc154421148)

[4. Вывод 13](#_Toc154421149)

# ВВЕДЕНИЕ

В малом бизнесе возникает потребность в программном обеспечении, в частности такие компании нуждаются в ПО для автоматизации бизнес-процессов, управления финансами, учета товаров и услуг, взаимодействия с клиентами и т.д. Взаимодействие с клиентами — это один из ключевых аспектов успешного бизнеса. В малом бизнесе для эффективного взаимодействия с клиентами могут использоваться различные инструменты и технологии, включая веб-приложения, мобильные решения. С этой целью могут использоваться различные каналы связи, такие как электронная почта, мессенджеры, социальные сети, телефон и т.д. Важно обеспечить быстрое и качественное реагирование на запросы и обращения клиентов, чтобы удовлетворить их потребности и сохранить положительное впечатление о компании.

Одним из элементов взаимодействия с клиентом является согласование с ним проведения работ или оказания ему услуг. Для этого требуется установить дату проведения соответствующих мероприятий. Это можно организовать путем личной встречи в офисе, либо с помощью мобильной связи, в частности с помощью обмена сообщениями в сети Интернет. Обмен сообщениями можно осуществить в социальной сети или в чуть менее мультисервисном решении – мессенджере. Однако, это требует монотонного повторения одних и тех же задач в общении с клиентом, процесс которых можно было бы автоматизировать. Для этого были разработаны сервисы по поиску мастеров и записи к ним на прием. Но такие сервисы имеют свои недостатки, которые необходимо исправить. Для этого необходимо разработать собственное приложение. Таким образом, целью данного проекта является проектирование и разработка сервиса по поиску мастера и записи к нему на прием. В соответствии с целью определены следующие задачи:

1. Провести анализ существующих решений;
2. Спроектировать и разработать сервис по поиску мастера и записи к нему на прием

# 1. Анализ существующих решений

Как правило, устоявшиеся на рынке небольшие компании, оказывающие определенные услуги, имеют либо свой сайт, где размещена информация об организации, услугах, прайс-лист, предоставлен функционал для приобретения услуги. Реже – помимо сайта имеют мобильное приложение с теми же функциями. Однако, эти программные продукты – индивидуальные решения, разработанные с учетом требований заказчика, со своими стилевыми и функциональными особенностями. В рамках данного проекта приложение будет общим для пользования всех заинтересованных лиц по согласованию приема. В качестве такого решения был рассмотрен сервис «ПИЛКИ» и «Rubitime».

***ПИЛКИ***

Данный сервис работает в виде мобильного приложения (рис. 1.1-1.2).

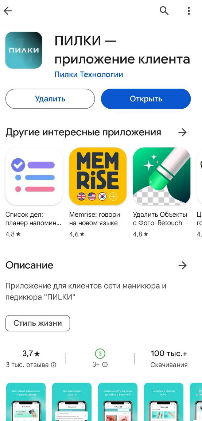


Рисунок 1.1 – Страница приложения в Google Play маркете

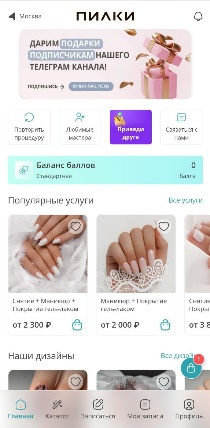


Рисунок 1.2 – Главная страница приложения

Решение предоставляет клиентский сервис для поиска мастеров по созданию и обслуживанию маникюра и бровей. Клиентам предлагается выбрать услугу либо мастера, так же существует выбор по разделам. Далее осуществляется выбор времени приема, мастера и студии. После определения клиентских критериев осуществляется поиск по актуальному местоположению, в результате чего клиенту выдается результат с подходящими мастерами или студиями. В конце клиенту предлагается записаться к выбранному мастеру в выбранное время (рис. 3)

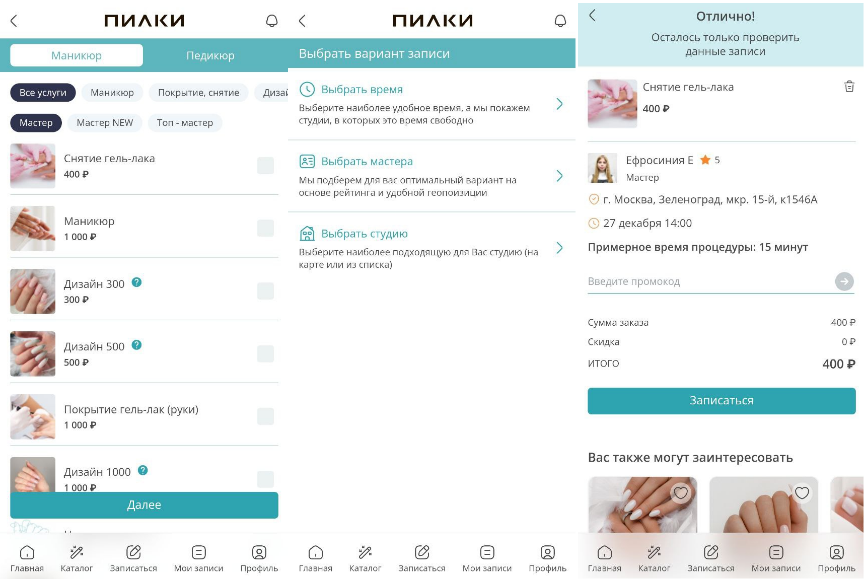


Рисунок 1.3 – Процесс поиска и записи к мастеру в приложении ПИЛКИ

Данное приложение осуществляет сбор информации из открытых источников и позволяет взаимодействовать с найденными мастерами.

***Rubitime***

Данное решение существует как веб-приложение, так и на мобильных платформах. Более широкий функционал, больше возможностей для клиентов, при этом данное ПО предназначено преимущественно для мастеров, которые нуждаются в организации своего плана малого бизнеса (рис. 4)



Рисунок 1.4 – Расписание на месяц в приложении Rubitime

В отличии от предыдущего это решение является платной, с возможностью пробного периода.

На основе рассмотренных решений была составлена их сравнительная характеристика (табл. 1.1)

Таблица 1.1 – Сравнительные характеристики

|  |  |  |
| --- | --- | --- |
| **Особенности** | ПИЛКИ | Rubitime |
| Платформа | Android | Мультиплатф. |
| Стоимость | Беспалтно | от 406 рублей в месяц |
| Целевой сегмент | Бьюти-индустрия (узкий) | Предприниматели, потребители (широкий) |
| Пользователи | Клиенты | Мастера и клиенты |
| Принцип работы | Парсинг открытых источников | Личная адаптация |

* *Платформа*. Более универсальным решением является «Rubitime», который работает на всех актуальных платформах;
* *Стоимость*. В отличии от бесплатного использования «ПИЛКИ» у «Rubitime» доступна лишь ограниченная бесплатная версия. Для использования более продвинутых функций и большего объема данных «Rubitime» требует ежемесячной оплаты подписки;
* *Целевой сегмент*. «ПИЛКИ» имеет только лишь две области для выбора мастера. «Rubitime» же в этом плане является универсальным;
* *Пользователи*. Пользователями «ПИЛКИ» являются только клиенты. «Rubitime» пользуются преимущественно мастера, но так же этим приложением могут пользоваться и клиенты;
* *Принцип работы*. «ПИЛКИ» не зависит от мастера, сервис самостоятельно подбирает коллекцию вариантов, а «Rubitime» требует настройки своей платформы от мастера.

# 2. Проектирование системы

## 2.1. Определение требований к системе

В первую очередь, приложение должно предназначаться как для мастеров, так и для клиентов. При этом надо учитывать то, что мастера работают в небольшой компании, поэтому предполагается, что мастер сам занимается своим планированием. В связи с этим, приложение будет актуально на мобильной платформе, так как это будет удобно и клиенту и мастеру. Однако, это может стать неудобством для новых клиентов, которые должны будут установить на свое устройство данное ПО. Для этого в качестве пользовательского интерфейса решено использовать чат-бот в Телеграм, что позволит пользователям без предварительных действий сразу начать пользоваться системой. Создание чат-бота предполагается осуществить с использованием библиотек мессенджера Телеграм, а именно с помощью telebot, так как данный API имеет более простую структуру, и позволяет сократить время на разработку. В итоге, определены следующие требования:

* Приложение в виде чат-бота Телеграм;
* Библиотека telebot;
* Предоставление управления мастерам и клиентам;
* Язык программирования Python.

## 2.2. Структура системы

Приложение должно обрабатывать клиентские запросы и предоставлять обратную связь. Для этого необходимо определить список вопросов от клиента и мастера. При этом необходимо выстроить очередь из вопросов, чтобы их обработка осуществлялась по порядку (табл. 2.2.1).

Таблица 2.2.1 – Структура запросов и ответов

|  |  |
| --- | --- |
| **Команда** | **Функция** |
| Первый шаг | |
| Мастер | Система определяет пользователя как мастера |
| Клиент | Система определяет пользователя как клиента |
| Второй шаг (клиент) | |
| Найти мастера | Система должна осуществить поиск мастера по никнейму, который по дальнейшей просьбе пользователь должен ввести |
| Текущие записи | Система предъявляет список текущих записей клиента |
| Второй шаг (мастер) | |
| Посмотреть расписание | Позволяет посмотреть текущее расписание, новые записи |
| Изменить расписание | Система предложит мастеру изменить текущее расписание или создать новое |
| Третий шаг (клиент) | |
| Текущий месяц | Выбор соответствующего месяца для записи на новый прием или просмотра имеющихся |
| Следующий месяц |
| Третий шаг (мастер) | |
| Текущий месяц | Выбор соответствующего месяца для просмотра расписания или его изменения |
| Следующий месяц |
| Четвертый шаг (клиент) | |
| Записаться | После выбора мастера и месяца система пишет список свободных дней мастера и предлагает записаться на него |
| Текущие | Возвращает записи на выбранный месяц |
| Четвертый шаг (мастер) | |
| Система принимает данные от мастера. Мастер вводит свободные дни | |

При начале сессии система должна будет записать данные пользователя: имя, никнейм, id. В процессе передачи запросов система так же должна вести учет очереди этих вопросов в пошагово в соответствии с таблицей 2.2.1.

Система должна иметь 3 логических раздела:

1. Основной. В этом разделе осуществляется логика бота
2. Менеджер данных. Необходимые операции с числовыми данными (дни месяца)
3. Пользователь. Здесь должны быть описаны функции каждого шага в соответствии с таблицей 2.2.1

# 3. Реализация разрабатываемой системы

***Менеджер данных (DataManager.py)***

В конструкторе класса определяется поле fileName, которому присваивается определенный путь.

В методе createMonth происходит создание расписания на месяц. В качестве параметров подаются userId – идентификатор пользователя, days – свободные дни, month – месяц (текущий или следующий). Описывается цикл от 1 (включ.) до 32 (не включительно), в котором шаг i (день) сравнивается с элементами списка days. Если i совпадает с одним из этих элементов, то создается словарь с ключами ‘day’, ‘time’ и ‘status’, где ‘status’ = y, что означает, что это свободный день. Иначе ‘status’ = no (день занят). Создается 31 строка, и данные подаются как объект с ключом month в json файл с соответствующим путем.

Метод getFreeDays возвращает список свободных дней. Считывается файл мастера по его пути, и загружается объект месяца month. Далее в цикле происходит поиск ‘status’ значение которого ‘y’. Если истина, то к образующемуся списку присоединяется день из этого словаря.

Метод add\_date служит для изменения статуса дня. По подаваемому дню и месяцу ищется соответствующая строка, в которой ‘status’ противопоставляется значение ‘y’ или ‘no’ или никнейм клиента

***Пользователь (User.py)***

В данном пакете создается родительский класс User, в котором определяются объект Telebot с токеном созданного бота, имя, никнейм, id.

От этого класса наследуется класс master и client. В этих классах описаны функции для каждой команды. Принцип их действия заключается в следующем: на вход подается объект message, который содержит текст сообщения и данные о пользователе. Открывается файл с текущей сессией пользователя. В нем содержатся 2 объекта: с данными о пользователе и коллекцией этапов (шагов). Достается объект с шагами и туда под ключом шага вносится значение введенного текста в предыдущем шаге. Создаются кнопки типа ReplyKeyboardMarkup с командами данного этапа. Далее из объекта Telebot вызывается send\_message для отправки сообщения пользователю с набором кнопок и текстом.

***Основной (BotTeleg.py)***

Данный класс начинается с создания объекта Telebot с передаваемым ему в параметры токеном и объект User, куда передается предыдущий объект.

Далее описываются два обработчика сообщений: команды ‘/start’ и другого текста. Это происходит с помощью декоратора функции bot.message\_handler. bot.message\_handler — это функция, которая принимает объект Message из Telegram API в качестве аргумента и вызывает соответствующую функцию обработки сообщения в зависимости от типа сообщения и других факторов. Это один из основных компонентов обработчика сообщений в боте Telegram, который позволяет боту реагировать на сообщения, отправленные пользователями.

При команде старт в User передаются данные о пользователе. Создается сессия, содержащие эти данные и передается в json-файл, который записывается в папку current\_sessions под названием, состоящим из Id этого пользователя в Телеграм. Описываются кнопки для следующих команд (рис. 3.1).

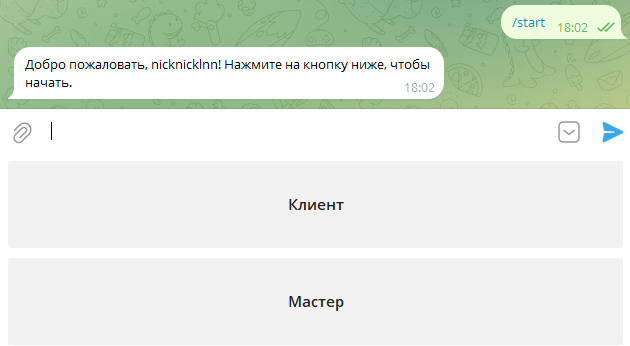


Рисунок 3.1 – Команда /start

В первом шаге определяется тип пользователя, в соответствии с которым создается определенный объект класса, наследующего User, появляется текст и кнопки для следующих команд второго шага (рис. 3.2)

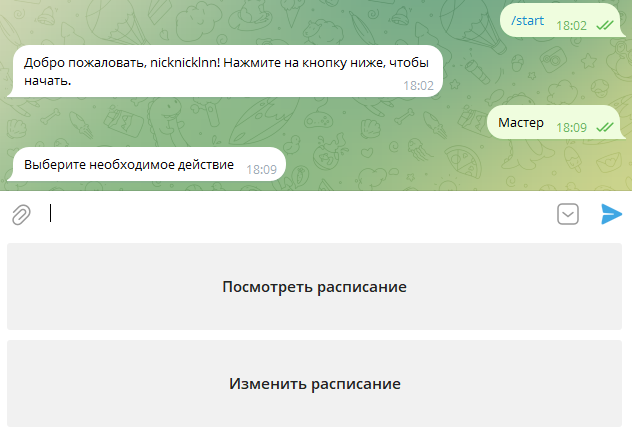


Рисунок 3.2 – Команда Мастер

Дальше предлагается изменить расписание, выбрав месяц. Если до этого расписание не было создано, создается новое (рис. 3.3)

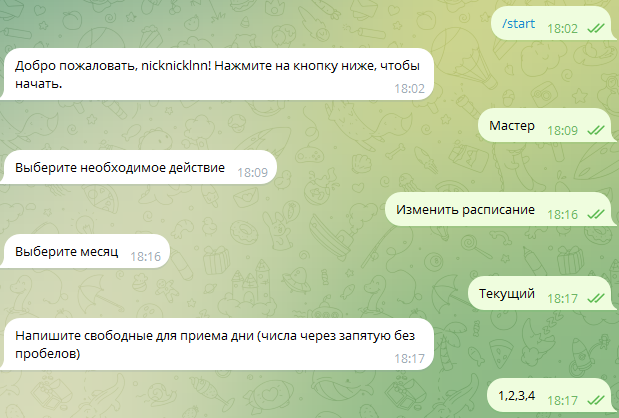


Рисунок 3.3 – Создание нового расписания на месяц

В соответствующей директории создается json-файл с расписанием (рис. 3.4).

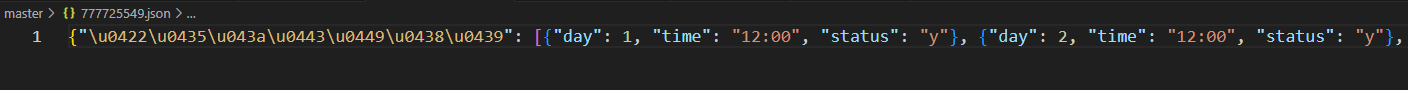


Рисунок 3.4 – Создание json-файла

# 4. Вывод

Обмен сообщениями можно осуществить в социальной сети или в чуть менее мультисервисном решении – мессенджере. Однако, это требует монотонного повторения одних и тех же задач в общении с клиентом, процесс которых можно было бы автоматизировать. Для этого были разработаны сервисы по поиску мастеров и записи к ним на прием. Но такие сервисы имеют свои недостатки, которых было лишено разработанное в ходе данной проектной работы приложение. Для этого был проведен анализ существующих решений, спроектирован и разработан сервис по поиску мастера и записи к нему на прием.