МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ПРОГРАММНОГО РЕШЕНИЯ ДЛЯ ОТДЕЛА ПО ЗАПУСКУ ПРОЕКТА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 Программная инженерия
факультета КНиИТ
Миронова Алексея Витальевича

Научный руководитель
доцент, к. ф.-м. н. _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Объектом исследования является проблема автоматизации целевых функций консультанта по запуску проекта.

Цель выпускной квалификационной работы – устранение проблем автоматизации предприятия путем разработки и внедрения программного решения. В ходе анализа деятельности предприятия была выявлена проблема отсутствия инструментария для взаимодействия консультантов отдела запуска проекта с системой планирования доставки, развернутой на предприятии. Проблема отсутствия данного инструментария является актуальной, так как для её решения тратятся существенные материальные ресурсы компании и ухудшается качество сервиса, предоставляемого клиентам.

Для решения проблемы работы с system delivery planning(длаее — SDP) системой были поставлены следующие задачи:

- Изучение ИТ инфраструктуры предприятия;
- Изучение процессов взаимодействия консультанта с ИТ системами;
- Разработка функциональных требований;
- Выбор технологий реализации программного решения;
- Проектирование программного решения;
- Реализация и внедрение программного решения.

Выпускная квалификационная работа содержит два раздела:

- Исследование предприятия и постановка задачи;
- Проектирование и реализация программного решения;

В разделе «Исследование предприятия и постановка задачи» приведена краткая характеристика предприятия, описание организационной структуры и инфраструктуры предприятия, проанализированы процессы работы консультанта по запуску проекта с системой планирования доставки, выведены функциональные требования к решению, произведён выбор литературы и технологий для реализации программного решения.

В разделе «Проектирование и реализация программного решения» описано проектирование модулей программного решения, представлена схема и взаимодействия программного решения с системой планирования доставки, описаны этапы разработки каждого из модулей, тестирование программы в целом и внедрение решения в продуктивную среду предприятия.

1 Теоретическая часть

1.1 Характеристика предприятия

В качестве объекта для исследования, выявления проблем и их решения, был выбран продуктивный контур ИТ департамента в компании ООО «Файв Пост».

Компания ООО «Файв Пост» является проектом компании «X5 Retail Group». Основной вид деятельности «Файв Пост» – грузоперевозки. Компания занимается доставкой товаров из интернет-магазинов и маркетплейсов до сетевых торговых точек «Пятерочка» и «Перекресток», а также в постаматные сети. На текущий момент «X5 Retail Group» обладает развитой логистической инфраструктурой, которая включает в себя более 40 распределительных центров, обеспечивающих более 12 000 магазинов в 7 федеральных округах России. Головной офис компании расположен в городе Москва по адресу улица Средняя Калитниковская, дом 28.

1.2 Описание деятельности отделов ИТ департамента

Разработкой ПО занимается Отдел Разработки «Файв Пост». При разработке используется гибкая методология управления проектами – Agile [1]. Разработка программных решения для внедрения нового функционала осуществляется сторонней компанией при помощи фреймворка Scrum. Все программные решения для «Файв Пост» разрабатываются и интегрируются итеративно [1]. Каждая итерация — sprint имеет фиксированное время исполнения и включает в себя полный список работ: от разработки технического задания, до программной реализации и демонстрации работы функционала. После каждого спринта отдел Development Operations интегрирует готовый функционал в продуктивную среду. Команды разработки сторонней компании находятся в постоянном контакте с владельцем продукта и scrum-мастером от компании Файв Пост. Владелец продукта отвечает за развитие продукта, составляет очередь задач для команды разработки при помощи историй, полученных от пользователей системы. Scrum-мастер отвечает за менеджмент команды разработки – проведение встреч и собраний, написание отчетов, повышение эффективности команды разработки [2]. Таким образом функционал внедряется в MVP постепенно.

1.3 Постановка задачи и разработка требований к решению

Для решения ранее проблем автоматизации, с которыми сталкивается консультант при обработке обращений, формировании отчетов и анализе метрик, предприятию необходимо автоматизированное рабочее место консультанта. Программное решение, далее АРМ — должно повысить качество и количество обработанных обращений, так же положительно повлиять на качество и скорость предоставления отчетов, тем самым освободив рабочее время консультанта и исключив операционные ошибки [3]. Так же АРМ должно предоставлять сведения в реальном времени о доступности микросервисной инфраструктуры предприятия, с возможностью отправлять оповещения консультанту при возникновении аварийных ситуаций. Программный продукт должен иметь возможность функционировать в операционной системе Microsoft Windows 10 и Microsoft Windows Server 2016. В конфигурации АРМ для отдела запуска проекта будут предусмотрены следующие функции:

- Получение информации о клиентских отправлениях;
- Изменение информации о клиентском отправлении;
- Запуск процессов в отношении клиентского отправления;
- Получение информации о плечах доставки клиентского отправления и их корректировка;
- Получение и изменение информации объектах ООО Файв Пост;
- Получение информации о связках между объектами и изменение связок;
- Получение информации о витрине партнера и изменение видимости точек выдачи на витрине партнеров;
- Логирование действий пользователей системы;
- Аутентификация пользователей.

Программа должна осуществлять поиск клиентских отправлений в системе ООО Файв Пост при помощи штрихкода, uuid заказа, uuid грузоместа и номеров клиентского отправления партнера и клиента, а также выводить следующую информацию о клиентском отправлении на экран:

- Данные заказа;
- Данные груза;
- Данные точки выдачи;
- Данные связанного с точкой выдачи магазина;
- Информация о задании на выдачу;

— Все процессы, запущенные в отношении клиентского отправления;
— Плечи доставки клиентского отправления;
— История статусов клиентского отправления.

Пользователь АРМ должен иметь возможность изменять информацию о клиентских отправлениях, при помощи взаимодействия с микросервисами.

Перечень характеристик клиентского отправления, которые пользователь иметь возможность изменить при помощи АРМ:

— Весогрузовые характеристики;
— Стоимость товара и доставки, оплата и тип оплаты;
— Точка выдачи, куда осуществляется доставка клиентского отправления;
— Изменение операционного статуса заказа;
— Изменение статуса груза;
— Изменение способа и параметра оплаты товара;
— Изменение системного местоположения клиентского отправления.

Доставка, выдача и возврат отправления осуществляются при помощи процессов в системе. Управление процессами осуществляется при помощи запросов к API микросервисов delivery и pickup_task. Пользователь АРМ должен иметь возможность запускать следующие процессы в отношении любого клиентского отправления:

— Процессы доставки – прямой, обратный, до склада первой мили, до склада последней мили;
— Процесс отмены процесса доставки любого типа;
— Процесс создания задания на выдачу;
— Процесс отмены задания на выдачу;
— Процесс принудительного системного изъятия клиентского отправления из точки выдачи;
— Процесс принудительной системной выдачи клиентского отправления;
— Процесс утилизации (списания) клиентского отправления;
— Процесс возврата партнеру клиентского отправления;
— Отмена процесса для плеча;
— Переотправка команды для создания приходной накладной в систему WMS для плеча;
— Переотправка команды для создания заказа на отгрузку в систему WMS для плеча.

Между складом партнера и точкой выдачи, есть дополнительные пункты загрузки и выгрузки. Расстояние между такими пунктами называется плечом доставки. Информация о плечах доставки для каждого клиентского отправления хранится в базе данных deliverydb. Для каждого плеча доставки запускается свой процесс, который формирует приходную накладную на склад поставки и после принятия посылки и размещения её в ячейку формируется заказ на отгрузку. АРМ должно позволить пользователю получать следующую информацию о плечах:

- Uuid плеча доставки;
- Статус плеча;
- Название отправителя;
- Uuid отправителя;
- Название получателя;
- Uuid получателя;
- Delivery Task ID.

Каждый из объектов сети X5 Retail Group, формирующих сеть доставки Файв Пост, имеет ряд характеристик, таких как: наименование, расположение, контакты для связи и т.д. Пользователь АРМ должен иметь возможность видеть на экране и изменять нижеописанные характеристики. Для склада и магазина:

- Uuid объекта;
- Тип объекта;
- Название;
- SAP-код;
- Nq-код;
- ФИАС-код;
- Макрорегион;
- Физический адрес;
- Координаты (широта и долгота);
- Контактный Email;
- Контактный телефон.

Для точки выдачи кассового типа:

- Uuid точки выдачи;
- Название;

— Внешний статус;
— Внутренний статус.

Для постамата сторонней компании:

— Uuid точки выдачи;
— Название;
— Внешний статус;
— Внутренний статус;
— Название партнера.

Для постамата сети «Файв Пост»:

— Uuid точки выдачи;
— Название;
— Внешний статус;
— Внутренний статус;
— Статус сети магазина;
— IP маршрутизатора магазина;
— IP коммутатора постамата;
— IP VPN шлюза;
— Номер порта коммутатора магазина, в который подключен постамат;
— Состояние порта;
— Правильность конфигурации порта;
— Номер VLAN.

Микросервис delivery планирует маршрут доставки клиентского отправления исходя из связок между объектами. Информация о связях между объектами хранится в базе данных routedb. Управление связками осуществляется при помощи транзакций в базу данных routedb. АРМ должно предоставить пользователю возможность просматривать, создавать, активировать и деактивировать связки.

Для того, чтобы клиенты партнера могли заказать товары при помощи Файв Пост партнеру необходимо передать определенный список точек, который в дальнейшем будет отображен на интернет-ресурсе партнера. АРМ должна позволить пользователю администрировать данную функцию, а именно:

— Выводить на экран список доступных точек для каждого партнера, а также предоставлять его в формате xlsx;

— Добавлять точки на витрину для выбранных партнеров;
— Убирать точки с витрины для определенных партнеров.

Управление витриной партнеров осуществляется при помощи транзакций в базу данных contractordb.

Все отчеты предоставляются при помощи корпоративной почты в формате xlsx. Информация для формирования отчетов хранится в базах, данных управляемых СУБД PostgreSQL. АРМ должна позволить консультанту выгружать информацию из базы данных, как по определенным заданным шаблонам, так и в зависимости от критериев, выбранных пользователем при помощи графического интерфейса. Выгрузка должна происходить на жесткий диск корпоративного компьютера либо на сетевое расположение терминального сервера.

Логирование – подразумевает запись действий пользователя в программе в централизованную базу данных на VDS сервере компании. Каждая запись должна обладать следующей информацией:

— Идентификатор пользователя;
— Взаимодействие с системой;
— Статус выполнения действия;
— Дата и время выполнения действия.

Аутентификация пользователя в АРМ необходима для разграничения уровня доступа пользователей к системе. Основные функциональные требования для аутентификации пользователя в АРМ это: самодостаточность аутентификации и валидации, а так же использование корпоративного продукта для авторизации.

1.4 Выбор технологий реализации

В качестве языка программирования для написания программного решения был выбран Python 3. Python 3 является мощным объектно-ориентированным языком программирования [4]. Решение использовать язык программирования Python 3 было принято в следствие следующих преимуществ языка программирования:

— Кроссплатформенность;
— Универсальность;
— Востребованность – из-за низкого порога вхождения и широкой области применения язык Python 3 имеет огромное сообщество программистов по всему миру. Соответственно найти ответ на интересующий вопрос

или необходимый для решения поставленной задачи модуль не составляет большого труда.

— Синтаксис – синтаксис языка Python 3 логичен и легкочитаем, что позволяет продуктивнее работать с программным кодом.

Данный язык программирования используется непосредственно разработчиками SDP-системы и тестировщиками. IDE и все необходимые библиотеки предустановлены на каждую корпоративную машину, предоставляемую сотрудникам информационно-технического отдела.

В качестве базы данных и СУБД на всех контурах (PROD, PRERPOD, DEV, TEST, STAGE) используется PostgreSQL. Оптимальным решением для работы с СУБД и формировании отчетов из полученных данных является использование модулей Python 3: openpyxl и psycopg2.

Openpyxl — модуль Python 3, предназначенный для чтения, записи и создания файлов формата Office Open XML, что полностью удовлетворяет функциональным требованиям формирования отчётов.

Psycopg2 — модуль Python 3, являющийся адаптером для СУБД PostgreSQL [5]. Адаптер позволяет отправлять запросы и транзакции в базу данных в несколько потоков и имеет возможность работать с кодировкой Unicode UTF-8. Все взаимодействия с микросервисами и платформами происходят с использованием технологии API. Python 3 имеет встроенный модуль для работы с json, а также отдельно загружаемую библиотеку requests, которую можно использовать для составления и отправки REST-запросов.

Json —- модуль языка программирования Python 3, который позволяет как декодировать данные из формата json в формат Python dictionary, так и кодировать данные из формата Python dictionary в формат json. Requests — модуль Python 3 для составления и отправки GET, PUT, POST, DELETE, PATCH, HEAD запросов к серверу [6].

В качестве средства для визуализации графического интерфейса была выбрана библиотека Qt5. Основные преимущества Qt5:

— Кроссплатформенность;

— Наличие IDE для проектировки интерфейсов;

— Удобный API.

Привязка инструментария Qt5 к языку программирования Python осуществляется при помощи пакета PySide2 [7].

Для серверной части приложения, необходимой для логирования действий и аутентификации пользователя в АРМ было принято решение использовать VDS компании. В качестве операционной системы будет использоваться Debian. Для функции аутентификации пользователя будет использоваться Red Hat KeyCloak. Это открытый проект для создания централизованной системы по управлению идентификацией пользователей, задания политик доступа и аудита для сетей на базе Linux и Unix. Основные преимущества использования KeyCloak:

- SSO при помощи единой учетной записи, используя Kerberos и LDAP для доступа к компьютерам и службам;
- Синхронизация с уже существующими серверами каталогов, например - Active Directory;
- Простота установки и управления законченным решением;
- Расширение функционала за счет подключаемых модулей.

В качестве хранилища для логов АРМ необходима система управления базами данных, имеющая возможность функционировать в Linux-подобных системах, иметь бесплатную лицензию, и быть совместимой с Python 3. Из всех доступных продуктов было принято решение использовать PostgreSQL — свободную объектно-реляционную систему управления базами данных.

АРМ будет работать с информацией из баз данных компании и полученными от микросервисов данными в формате JSON, а также АРМ не подразумевает хранение какой-либо постоянной информации на клиентской стороне. От встроенной в клиентскую часть программы базы данных было принято решение отказаться. Использование встроенной базы данных является избыточным решением, так как максимальный совокупный объем хранимых программой данных не превышает 10 мегабайт. Все данные уже имеют структурированы в формат JSON, для их хранения целесообразно использовать хэш-таблицы и оперативную память клиентской машины.

2 Практическая часть

Этап практической реализации включает в себя настройку СУБД PostgreSQL и средства авторизации Red Hat KeyCloak для серверной части программы, создание xml-разметки для графического интерфейса при помощи фреймворка Qt5 и написание программного кода на языке программирования Python 3.

Создание и настройка базы данных происходит на локальной машине, с последующим внедрением в продуктивный контур предприятия. Для управления СУБД используется имеющееся в дистрибутиве PostgreSQL 13 средство администрирования – psql. Согласно проекту, необходимо создать базу данных loggerdb со структурой таблиц, указанной на рисунке 1.1. Перед созданием базы данных, необходимо создать двух пользователей – владельца и обозревателя. В Приложении А представлен SQL-запрос для создания супер-пользователя и пользователя с правами обозревателя.

После того, когда все пользователи созданы, необходимо повторно подключиться к серверу PostgreSQL под пользователем omniowner и создать базу данных loggerdb и передать на нёё права пользователям. В данном случае omniowner получает права суперпользователя, а omniviewer права на подключение к базе данных.

При разработке графического интерфейса, использовалась среда разработки Qt Designer, которая входит в состав пакета PySide2 для Python 3 [8]. Qt Designer позволяет создавать интерфейсы в режиме «что видишь – то получишь» (WYSIWYG), содержит полный набор виджетов Qt и позволяет компоновать их и редактировать их свойства.

Для ускорения процесса разработки создается отдельный файл с шаблонами часто повторяющихся элементов интерфейса. Подобное решение позволит без труда привести все элементы интерфейса к одному стилю и расположить их внутри необходимых окон [9].

Согласно проекту, графический интерфейс включает в себя пять окон – одно главное и четыре диалоговых. Создание графического интерфейса окна целесообразно начать с размещения виджетов для управления окном. На рисунке 16. изображено главное окно с виджетами для управления.

Для управления виджетами окна при помощи Python 3 необходимо сгенерировать Python класс из xml разметки окна. Данное действие производится при помощи команды:

```
«pyside2-uic "window.ui" -o "window.py"»
```

Результатом работы команды является файл «window.py», который содержит класс Ui_window, включающий в себя два метода: «setupUi» и «retranslateUi». Для настройки интерфейса окна необходимо создать новый класс, наследующий суперкласс PySide2, который будет инициализировать объекта класса Ui_MainWindow и обращаться к методу setupUi. В программном коде Python 3 эти действия записываются следующим образом:

```
from Ui_MainWindow import Ui_MainWindow
#  Создание класса MainWindow, наследующим QMainWindow
class MainWindow(QMainWindow):
    def __init__(self):
        #  Инициализация всех методов и переменных суперкласса
        QMainWindow.__init__(self)
        #  Создание объекта класса Ui_MainWindow
        self.ui = Ui_MainWindow()
        #  Обращение к методу setupUi класса Ui_MainWindow
        self.ui.setupUi(self)
```

Таким образом для каждого окна создается отдельный класс необходимый для настройки интерфейса. На рисунке **??** представлена схема классов для настройки интерфейса и их методов.

Окончательным этапом разработки GUI будет вызов диалоговых окон по нажатию кнопок в главном окне. Вызвать диалоговое окно, можно, инициализировав объекта класса-настройщика этого окна методом «exec_». Например, окно
Ui_DeliveryDialog, необходимое для создания процесса доставки, можно вызвать следующим образом:

```
self.delivery = DeliveryDialog
self.ui.createDeliveryButton.clicked.connect(lambda: self.delivery().exec_()
    )
```

Конечным этапом в реализации программной части является разработка модуля для работы с серверной частью. Он необходим для записи информации о действиях пользователя в базу данных и авторизации в приложении. Для авторизации используется метод check_auth класса AuthorisationWindow. Для авторизации пользователя в APM метод check_auth отправляет POST запрос с токеном «Bearer» и телом JSON, в котором указываются следующие параметры:

— client_id — id приложения в Red Hat;

— grant_type — тип авторизации;
— client_secret — секрет приложения в Red Hat;
— scope — протокол проверки подлинности;
— username — имя пользователя;
— password — пароль.

В случае успешной авторизации сервер возвращает токен JWT и код-ответа 200 [6], иначе информацию об ошибке авторизации и код-ответа 401.

Работа с логированием осуществляется при помощи Python 3 класса Logger. Каждый класс АРМ содержит объект этого класса, используемый для передачи информации о действии пользователя. Класс logger состоит из следующих методов:

— __init__ — конструктор класса, содержит объект подключения к серверной базе данных;
— user — метод для работы с таблицей user_t. Создает запись о пользователе либо обновляет столбец last_login_date текущим значением даты и времени на момент авторизации пользователя в системе;
— log_auth — метод, работающий с таблицей authlog_t, заполняет информацию о попытках авторизации пользователя;
— log_package — метод, работающий с таблицей packagelog_t, заполняет информацию о действиях пользователя с посылками;
— log_object — метод для работы с таблицей objectlog_t, заполняет информацию о действиях пользователя с объектами;
— log_contractor — метод для работы с таблицей contractorpointlog_t, заполняет информацию о действия пользователя с привязкой точек выдачи к партнеру (витрина партнеров);
— log_report — метод для работы с таблицей reportlog_t, заполняет информацию о действия пользователя с отчетами;
— close — метод, закрывающий соединение с базой данных loggerdb.

Каждый из методов, за исключением метода close, получает в качестве аргумента хэш-таблицу в виде словаря Python 3 с необходимыми параметрами для заполнения в базу данных. На рисунке 1 указана схема взаимодействия классов модуля для работы с SDP-системой.

Для удобства использования и распространения АРМ было принято решение упаковать исходный код, интерпретатор Python 3 и все зависимости в

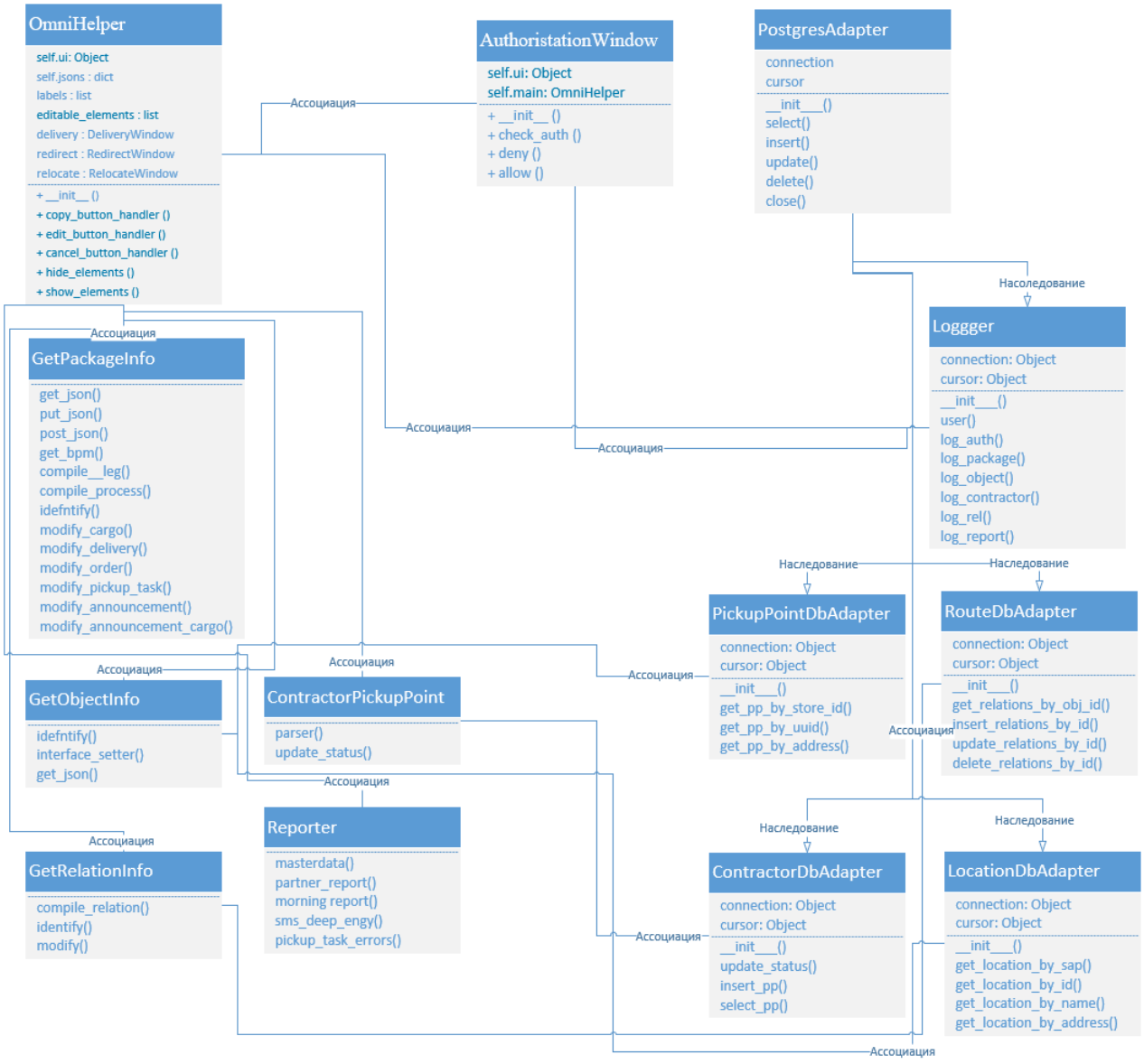

Рисунок 1 – Схема взаимодействия классов модуля для работы с SDP-системой

исполняемый файл [10]. Данные действия осуществлялись при помощи пакета PyInstaller. Основное преимущество этого пакета в том, что конечный пользователь может запускать приложение без установки интерпретатора Python 3 или каких-либо модулей. Готовый дистрибутив программы уже содержит виртуальную среду Python 3 и все зависимости. Установка PyInstaller происходит через менеджера пакетов Python 3 – pip:

```
pip install pyinstaller
```

Для упаковки исходного кода в приложение для операционных систем Microsoft Window использовалась следующая команда:

```
pyinstaller main.py
```

После исполнения данной команды, создается новая директория с исполняемым файлом, который можно распространить через корпоративные ресурсы всем сотрудникам отдела по запуску проекта [11].

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы была изучена проблема работы консультанта с системой планирования доставки, а именно: ручная обработка клиентских отправлений, объектов сети доставки и ручная выгрузка отчетов из базы данных. В прошлом проблема отсутствия инструментария для работы с SDP-системой решалась при помощи привлечения отдела по запуску проекта и найма сотрудников сторонних компаний. Данное решение проблемы не является достаточным и оптимальным для компании, так как несет за собой материальные убытки и отрицательные отзывы клиентов о компании.

На текущий момент, используя АРМ консультанта по запуску проекта, удалось достичь увеличения скорости обработки клиентских отправлений и работы с объектами сети доставки, за счет удобного и понятного графического интерфейса, и готовых алгоритмов по управлению сущностями SDP-системы. Количество ошибок при работе с системой планирования доставки было снижено, так как все алгоритмы программного решения были успешно протестированы и не подразумевают возможности изменения пользователем. Уровень подготовки и квалификации консультантов отдела по запуску проекта также возрос, в следствии наличия в АРМ, функции логирования действий пользователя. Использование информации о неверных действиях пользователя помогает руководителю оперативно объяснить пользователю некорректность его действий и оперативно их исправить.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Мартин, Р.* Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — СПб: Питер, 2009. — С. 352.

2 *Кон, М.* Пользовательские истории: гибкая разработка программного обеспечения / М. Кон. — Москва: Вильямс, 2012. — С. 256.

3 *Репин, Р.* Бизнес-процессы, моделирование, внедрение, управление / Р. Репин. — Москва: МиФ, 2014. — С. 512.

4 *Любанович, Б.* Простой Python. Современный стиль программирования / Б. Любанович. — СПб: Питер, 2019. — С. 592.

5 *Уорсли, Д.* PostgreSQL. Для профессионалов / Д. Уорсли, Д. Дрейк. — СПб: БХВ-Петербург, 2018. — С. 446.

6 *Subramanian, H.* Hands-On RESTful API Design Patterns and Best Practices / H. Subramanian, P. Raj. — UK, Birmingham: Packt Publishing, 2019. — P. 310.

7 *Лутц, М.* Изучаем Python. 4-е издание / М. Лутц. — СПб: Символ-Плюс, 2017. — С. 1280.

8 *Хахаев, И. А.* Практикум по алгоритмизации и программированию на Python[Электронная книга] / И. А. Хахаев. — Москва: НОУ «Интуит», 2016.

9 *Дунаев, В.* HTML, скрипты и стили / В. Дунаев. — СПб: БХВ-Петербург, 2018. — С. 1024.

10 *Цуканова, О.* Методология и инструментарий моделирования бизнес–процессов / О. Цуканова. — СПб: Университет ИТМО, 2015. — С. 100.

11 *Доусон, М.* Программируем на Python. / М. Доусон. — СПб: Питер, 2018. — С. 416.