

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное
Образовательное учреждение высшего образования
«Уфимский университет науки и технологий»

Факультет информатики и робототехники

Кафедра ВМиК

Отчет по лабораторной работе №3
по дисциплине «Проектирование и конструирование ПО»
на тему: ««Методология объектно-ориентированного моделирования.
Этап создания физической модели»

Выполнили:

Студенты группы ПРО-233Б

Бердин Д.С.

Терегулов Т.Р.

Чуриков М.А.

Проверил:

Преподаватель

Насыров Р.В.

Уфа - 2023

Лабораторная работа №3

1. Цель работы:

Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML.

Задание:

1. Определиться с диаграммами из семейства UML моделей на этапе создания физической модели автоматизированной системы.
2. Разработать UML диаграммы этапа создания физической модели.
3. Задokumentировать прецеденты.

Типичное описание должно содержать следующие разделы.

- a. Краткое описание.
 - b. Участвующие субъекты.
 - c. Предусловия, необходимые для инициирования прецедента.
 - d. Детализированное описание потока событий, которое включает: основной поток, который можно разбить для того, чтобы показать подчиненные потоки событий (подчиненные потоки могут быть разделены дальше на еще более мелкие потоки, с целью сделать читаемость документа более удобной); альтернативные потоки для определения исключительных ситуаций.
 - e. Постусловия, определяющие состояние системы, по достижении которых прецедент завершается.
4. Записать не менее 10 требований согласно синтаксиса требований: [обстоятельства] [субъект] [действие] [объект] [ограничение].

Пример: Когда сигнал получен [обстоятельства] система [субъект] должна установить [действие] разряд сигнала [объект] в течение двух секунд [ограничение].

5. Сформулировать нефункциональные требования.
6. Разработать алгоритм обработки данных.
 - a. Алгоритм обработки должен быть реализован с помощью объектно-ориентированного подхода. Обработчик данных реализуется в отдельном классе (**DataProcessor**), который имеет 3 базовых метода: чтение источника данных (**read**), запуск обработки данных (**run**), вывод результата на экран (**print_result**).
 - b. Реализовать в классе необходимые методы обработки (например, очистка, назначение категорий и т.п.).

- с. Использовать паттерн «Фабрика» (Factory) для вызова различных экземпляров `DataProcessor` в зависимости от типа входного набора данных (например, csv-файл, txt-файл).
 - d. Загрузить код на **GitHub**
 - e. Разработать UML схему классов обработчика данных.
7. **Написать отчет.** Отчет должен включать комплекс статических и динамических моделей, описание алгоритма обработки данных, скриншоты с результатами выполнения алгоритма (фрагмент входного набора данных и вывод после выполнения обработки), скриншот GitHub-репозитория с загруженными файлами проекта.

Ход работы

Нефункциональные требования:

1. Нefункциональные требования

1.1.Интерфейс пользователя

1.1.1. Интерфейс пользователя должен быть на русском языке.

1.1.2. Система должна отображать корректно интерфейс Пользователя с разрешением от 1024x600 пикселей.

1.2.Поддержка браузеров

1.2.1. Система должна работать для следующих браузеров последних версий:

MS Internet Explorer,

Mozilla Firefox, Google Chrome, Safari, Opera.

1.3.Требования к производительности

1.3.1. Система должна стабильно работать с глубиной истории не менее чем в 3 года.

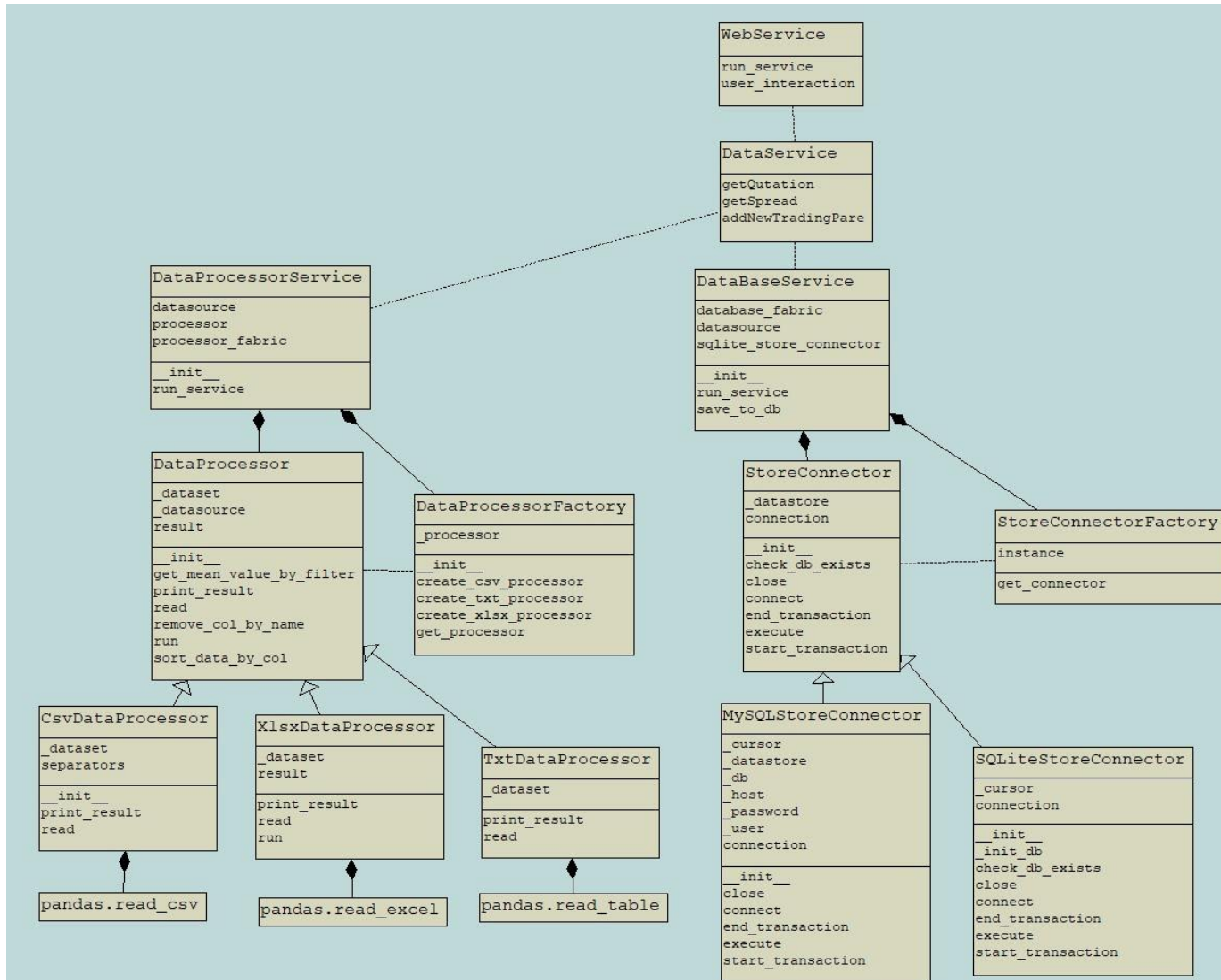
1.3.2. Система должна стабильно работать при 500 одновременно подключенных пользователей.

1.3.3. Система должна загружать любую форму не дольше, чем 5 секунд.

1.4.Требования к безопасности

1.4.1. Система НЕ должна позволять НЕ Администраторам физический доступ к интерфейсу администратора.

UML-диаграмма классов



Factory

```
dataprocessor.py x dataprocessor_factory.py x dataprocessor_service.py x
17
18 """
19 Фабричный метод может не только возвращать класс соответствующего обработчика,
20 здесь также может быть реализована логика, которая меняет поведение данного обработчика,
21 например, меняет тип разделителя и кодировку для CSV-файла (через атрибуты класса),
22 применяет различные режимы обработки и т.д.
23 """
24
25 def get_processor(self, datasource: str) -> Optional[DataProcessor]:
26     """ Основной фабричный метод, возвращающий необходимый объект класса DataProcessor
27     в зависимости от расширения файла """
28     if datasource.endswith('.csv'):
29         self.create_csv_processor(datasource)
30     elif datasource.endswith('.txt'):
31         self.create_txt_processor(datasource)
32     elif datasource.endswith('.xlsx'):
33         self.create_xlsx_processor(datasource)
34     return self._processor
35
36 def create_txt_processor(self, datasource: str) -> None:
37     """ Создаем TxtDataProcessor и пытаемся прочитать данные, если успешно, сохраняем объект в атрибуте класса """
38     processor = TxtDataProcessor(datasource)
39     if processor.read():
40         self._processor = processor
41
42 def create_csv_processor(self, datasource: str) -> None:
43     """ Создаем CsvDataProcessor и пытаемся прочитать данные, если успешно, сохраняем объект в атрибуте класса """
44     processor = CsvDataProcessor(datasource)
45     if processor.read():
46         self._processor = processor
47
48 def create_xlsx_processor(self, datasource: str) -> None:
49     """ Создаем XlsxDataProcessor и пытаемся прочитать данные, если успешно, сохраняем объект в атрибуте класса """
50     processor = XlsxDataProcessor(datasource)
51     if processor.read():
52         self._processor = processor
```

Методы обработки данных

```
def sort_data_by_col(self, df: pandas.DataFrame, colname: str, asc: bool) -> pandas.DataFrame:
    """
    Метод sort_data_by_col просто сортирует входной датасет по наименованию
    заданной колонки (аргумент colname) и устанавливает тип сортировки:
    ascending = True - по возрастанию, ascending = False - по убыванию
    """
    return df.sort_values(by=[colname], ascending=asc)

def remove_col_by_name(self, df: pandas.DataFrame, col_name: List[str]):
    """
    Метод remove_col_by_name принимает входной набор данных и список имён колонок для удаления.
    Возвращает набор данных с удалёнными колонками.
    """
    return df.drop(col_name, axis=1)

def get_mean_value_by_filter(self, df: pandas.DataFrame, filter_expr: str) -> pandas.DataFrame:
    """
    Метод get_mean_value_by_filter выбирает из входного набора данных строки с заданным условием
    (фильтр), используя инструкцию DataFrame.query(), применяет к получившимся значениям функцию
    mean (считает среднее значения в колонках) и возвращает результат в виде нового DataFrame.

    Подробнее по фильтрации строк с помощью query() см.: https://sparkbyexamples.com/pandas/pandas-filter-by-column-value/
    """
    result = df.query(filter_expr)
    return result.mean(axis=0, skipna=True).to_frame().T

@abstractmethod
def print_result(self) -> None:
    """ Абстрактный метод для вывода результата на экран """
    pass
```

Обработчик Excel-файлов

```
dataprocessor.py
dataprocessor.py x
121
122
123 class TxtDataProcessor(DataProcessor):...
140
141
142 class XlsxDataProcessor(DataProcessor):
143     """ Реализация класса-обработчика Excel-файлов """
144
145     def read(self):
146         """ Реализация метода для чтения Excel-файла """
147         try:
148             self._dataset = pandas.read_excel(self._datasource)
149             col_names = self._dataset.columns
150             if len(col_names) < 2:
151                 return False
152             return True
153         except Exception as e:
154             print(str(e))
155             return False
156
157     def run(self):
158         self.result = pandas.DataFrame()
159         # удаляем ненужные колонки из входного набора данных
160         cleaned_result = self.remove_col_by_name(self._dataset, ["xd"])
161         target_values = ["count", "price"] # значения колонки "target" для фильтрации
162         mean_result = 0
163         for t in target_values:
164             # фильтруем строки по каждому возможному значению колонки "target"
165             mean_result = self.sort_data_by_col(cleaned_result, t, False)
166
167         # полученный результат (DataFrame) добавляем в результирующий DataFrame в self.result
168         self.result = pandas.concat([self.result, mean_result], ignore_index=True)
169
170     def print_result(self):
171         print(f'Running XLSX-file processor!\n', self.result)
172
```

Алгоритм обработки данных

Метод **run()** реализует обработку данных. Он удаляет ненужные столбцы с датами при помощи метода **remove_col_by_name(df DataFrame, filter [])**. Затем с помощью функции **sort_data_by_col(df DataFrame, colname, asc)** сортирует входной датасет по наименованию заданной колонки (аргумент *colname*) и устанавливает тип сортировки: **ascending = True** – по возрастанию, **ascending = False** – по убыванию.

Обработанные данные

```
Running XLSX-file processor!
```

	count	Exchange	asset1	asset2	price
0	1567	Gate.io	WBTC	TRY	30555.88000
1	20	Binance	BTC	TRY	30481.97000
2	278	KuCoin	ROOBEE	BTC	30166.33000
3	70	Binance	BTC	ZAR	28287.75000
4	178	KuCoin	VSYS	BTC	28106.80000
...
5640	1307	Poloniex	SHIB	USDC	0.00001
5641	1287	Bithumb	SHIB	KRW	0.00001
5642	1345	ShibaSwap	SHIB	BONE	0.00001
5643	1310	Huobi	SHIB	USDD	0.00001
5644	1357	BitTurk	SHIB	TRY	0.00001

[5645 rows x 5 columns]

Прецеденты

Прецедент	Файл не найден
Краткое описание	Данный прецедент необходим для сообщения администратору о необходимости устранения ошибок в случае неполадок с файлом.
Субъект	Система, оператор
Основной поток	Система сообщает оператору об ошибке. Запросы пользователей на подсчет спреда временно блокируются.
Альтернативный поток	Если сервер не отвечает, администратору также сообщается об этом.
Постусловия	После завершения прецедента, т.е. устранения ошибки, система функционирует в прежнем режиме.

Прецедент	Запрос на получение спреда
Краткое описание	Данный прецедент необходим для получения спреда клиентом
Субъект	Система, клиент
Основной поток	Пользователь по запросу на сервер может получить информацию о наличии спредов и конкретные спреды
Альтернативный поток	Если сервер не отвечает, пользователю сообщается о временных технических неполадках
Постусловия	После успешного завершения прецедента спреды будут отображены на сайте и занесены в базу данных

Прецедент	Добавление торговой пары
Краткое описание	Данный прецедент необходим для добавления в файл новых торговых пар.
Субъект	Администратор, система
Предусловия	Администратор должен заполнить таблицу новых торговых пар
Основной поток	Администратор делает запрос на добавление новой записи файл.
Постусловия	После успешного завершения прецедента данные о новой торговой паре внесены в файл.

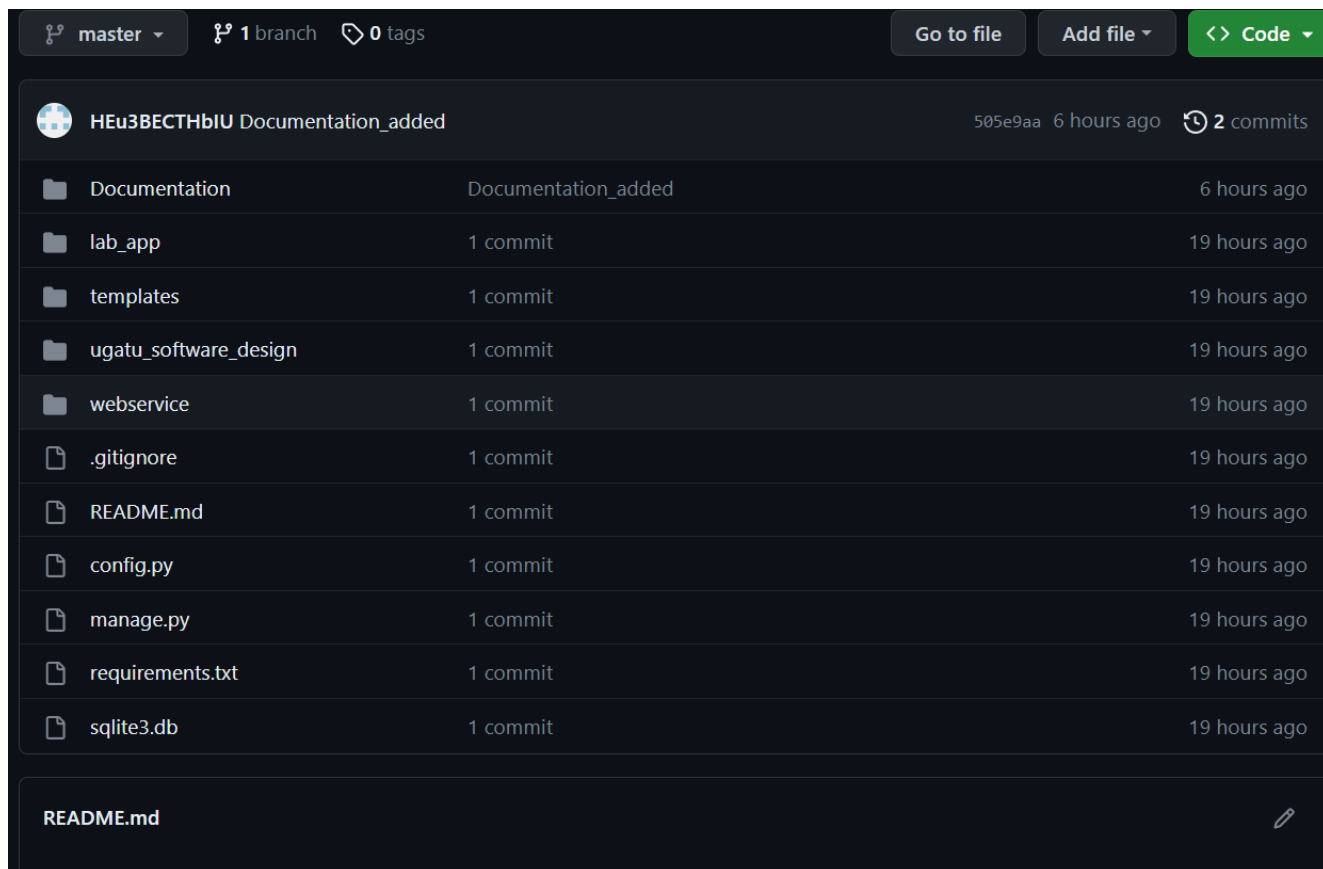
Прецедент	Парсинг торговой пары
Краткое описание	Данный прецедент необходим для парсинга данных из файла в БД.
Субъект	Администратор, система, файл
Предусловия	Файл должен существовать и путь до него должен быть указан корректно
Основной поток	Система обрабатывает файл построчно и заносит обработанные данные в БД
Альтернативный поток	Если текущая строка некорректна, то администратору выдается сообщение об ошибке
Постусловия	После успешного завершения прецедента данные обо всех торговых парах занесены в БД

Прецедент	Регистрация пользователя
Краткое описание	Данный прецедент необходим для регистрации пользователя в системе.
Субъект	Система, незарегистрированный пользователь
Предусловия	Пользователь должен открыть форму для регистрации
Основной поток	Пользователь заполняет форму и отправляет запрос на регистрацию в системе
Альтернативный поток	Если пользователь ввел некорректные данные или не заполнил обязательные поля, выдается сообщение об ошибке и предлагается заполнить форму еще раз
Постусловия	После успешного завершения прецедента данные о новом пользователе заносятся в базу данных.

Прецедент	Вход пользователя
Краткое описание	Данный прецедент необходим для входа пользователя в систему.
Субъект	Система, пользователь
Предусловия	Пользователь должен открыть форму для входа
Основной поток	Пользователь заполняет форму и отправляет запрос на вход в систему
Альтернативный поток	Если пользователь ввел некорректные данные (таких данных нет в БД) или не заполнил обязательные поля, выдается сообщение об ошибке и предлагается выполнить вход еще раз
Постусловия	После успешного завершения прецедента пользователь входит в систему.

Скриншот репозитория с загруженными файлами проекта

Ссылка: <https://github.com/sadchordoma/ugatu-software-design>



Выводы к работе.

В ходе выполнения лабораторной работы мы ознакомились с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML, а также разработали UML модели для реализации нашей автоматизированной системы.