

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 № 4
на тему: «Проектирование БД. Проектирование пользовательского
интерфейса»

Выполнили:

Студенты группы ПРО-233Б

Бердин Д. С.

Чуриков М. А.

Терегулов Т. Р.

Проверил:

Ст. преподаватель

Насыров Р. В.

Уфа - 2023

Лабораторная работа №4

Цель работы:

- Ознакомление с основными методами проектирования базы данных (БД).
- Ознакомление с методами проектирования пользовательского интерфейса.

Задание:

Проектирование БД

1. Изучить дополнительный материал по теории проектирования БД. Спроектировать БД приложения для хранения обработанных данных.
 - а. БД должна содержать таблицу с информацией об обработанных файлах (минимальный набор колонок: название файла, дата обработки).
 - б. Таблица обработанных файлов должна ограничивать основную таблицу с данными по внешнему ключу (FOREIGN KEY).
2. Написать SQL-скрипт для создания структуры спроектированной БД.
3. Изучить код примера из архива `piкpo4_python.zip`. На основе данного примера реализовать необходимые CRUD (Create, Read, Update, Delete) операции для работы с БД (см. ссылку на материал по основам SQL). Загрузить код приложения на GitHub.
4. Проверить выполнение CRUD-операций на тестовой БД (только SQLite).

Проектирование пользовательского интерфейса

Во время выполнения лабораторной работы необходимо описать ожидаемое поведение разрабатываемой системы с точки зрения внешнего по отношению к ней пользователя, то есть осуществить

"конструирование" внешних взаимодействий будущей ИС с пользователем без конкретизации его внутреннего устройства.

1. Определить структуру проектируемого пользовательского интерфейса (визуальное оформление, отвечающее за представление информации пользователю; функциональные возможности системы, включающие набор возможностей для эффективного выполнения профессиональной деятельности; техники взаимодействия пользователя с системой + дополнительные функциональные возможности системы) с учетом задачи (лекции, рекомендации Приложение 1, дополнительные функциональные возможности системы (пример) в Приложении 3).

2. Определить стили пользовательского интерфейса (графический (GUI, web-интерфейс (WUI), объектно-ориентированный интерфейс) с учетом задачи (лекции, рекомендации Приложение 1).

3. Определиться с размещением элементов пользовательского интерфейса (кнопки, иконки, выпадающие списки, поля для записи текста и пр.) (Лекции, рекомендации Приложение 1).

4. Написать Требования к интерфейсу пользователя (Пример в приложении 2).

5. Разработать взаимодействие разрабатываемой программы с пользователем: сценарий (можно диаграммой последовательности, диаграммой взаимодействия), экранные формы, набор подсказок (перечисление), и пр.

6. Ознакомиться с методическим материалом по базовой верстке веб-страниц.

7. На основе п.1-5 разработать (сверстать) основные html-страницы для вашего приложения, используя CSS-стили и HTML5-модель верстки (см. пример реализации HTML5-страниц в папке www). Загрузить код html страниц на GitHub.

Написать отчет. Отчет должен содержать:

- a. ER-диаграмму спроектированной БД.
- b. Листинг SQL-скрипта для создания структуры БД.
- c. Листинг основных функций для работы с БД (с пояснениями, что делает каждая функция).
- d. Скриншот части содержимого основной таблицы данных и связанных таблиц.

Скриншоты основных html-страниц с пояснениями и обозначением разметки базовых HTML5-тегов.

Ход работы

Взаимодействие разрабатываемой программы с пользователем

Взаимодействие с незарегистрированным пользователем осуществляется через сайт, посредством нажатия на кнопку «Shunter», которая выводит топ-100 крипто активов по капитализации. Кнопка «Topic» выводит статью для ознакомления с понятием «спред». Также у незарегистрированного пользователя есть возможность зарегистрироваться (кнопка «Login») для получения доступа к функционалу пользователя.

Функционал пользователя дополнительно включает возможность отображения текущих спредов по нажатию на кнопку «Spread Today».

Требования к пользовательскому интерфейсу:

1. Простота интерфейса
2. Интуитивность интерфейса
3. Пригодность к использованию
4. Понимание интерфейса пользователем
5. Масштабируемость интерфейса

ER- диаграмма БД

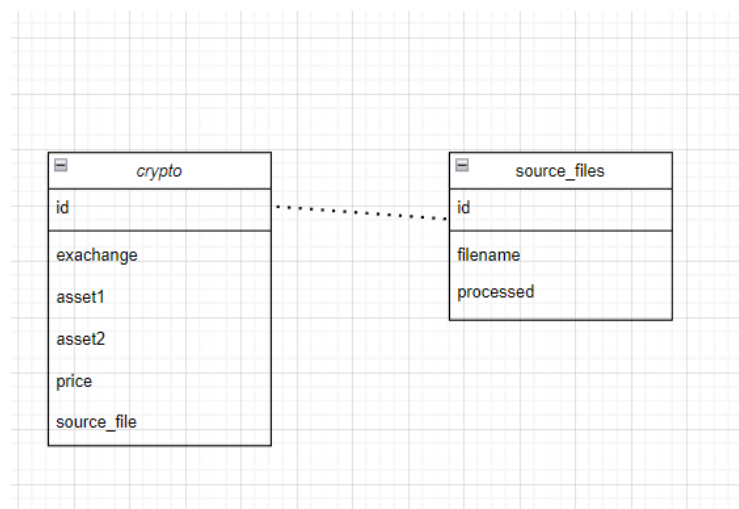


Рисунок 1 ER-диаграмма

Скриншоты части содержимого основной таблицы данных и связанных таблиц.

sqlite3.db

Tables Database Metadata

Table: crypto

Page: Jump << < 1-113 > >> Refresh

id	exchange	asset1	asset2	price	source_file
1	Gate.io	WBTC	TRY	30555.88	1
2	Binance	BTC	TRY	30481.97	1
3	KuCoin	ROOBEE	BTC	30166.33	1
4	Binance	BTC	ZAR	28287.75	1
5	KuCoin	VSYS	BTC	28106.8	1
6	Bittrex	WBTC	USDT	28065.22	1
7	Binance	BTC	RUB	28054.22	1
8	Binance	BTC	UAH	27895.67	1
9	Binance	BTC	BRL	27741.8	1
10	Balancer	WBTC	WETH	27670.39	1
11	Bitfinex	BTC	EUR	27654.7	1
12	Coinbase	BTC	EUR	27632.84	1
13	Kraken	XBT	EUR	27632.73	1
14	Bitstamp	BTC	EUR	27630.29	1
15	Binance	BTC	EUR	27627.67	1
16	Kraken	WBTC	EUR	27618.08	1
17	KuCoin	DRGN	BTC	27601.57	1
18	Binance	BTC	PLN	27522.32	1
19	KuCoin	BTC	TUSD	27506.11	1
20	Binance	BTC	GBP	27488.9	1
21	Kraken	XBT	CHF	27436.75	1
22	Kraken	XBT	AUD	27405.51	1
23	Binance	BTC	AUD	27392.29	1
24	SpookySwap	WFTM	WBTC	27387.2	1
25	SpiritSwap	WFTM	WBTC	27380.35	1
26	SpookySwap	USDC	WBTC	27379.31	1
27	Kraken	TRX	XBT	27377.49	1
28	SpookySwap	USDT	WBTC	27376.7	1
29	Solidly	WFTM	WBTC	27370.79	1
30	Bitstamp	WBTC	BTC	27369.55	1
31	Bitfinex	BTC	GBP	27353.26	1

sqlite3.db

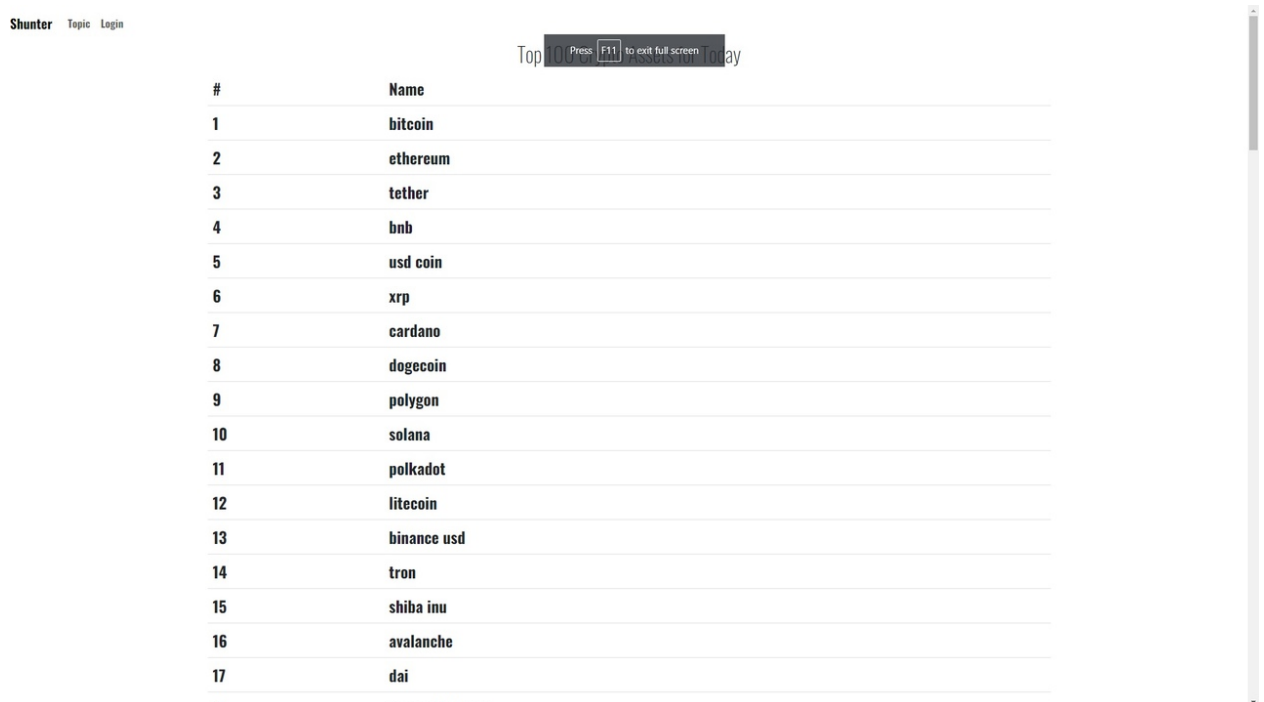
Tables Database Metadata

Table: source_files

Page: Jump << < 1-1 > >> Refresh

id	filename	processed
1	crypto_ma...	2023-04-23...

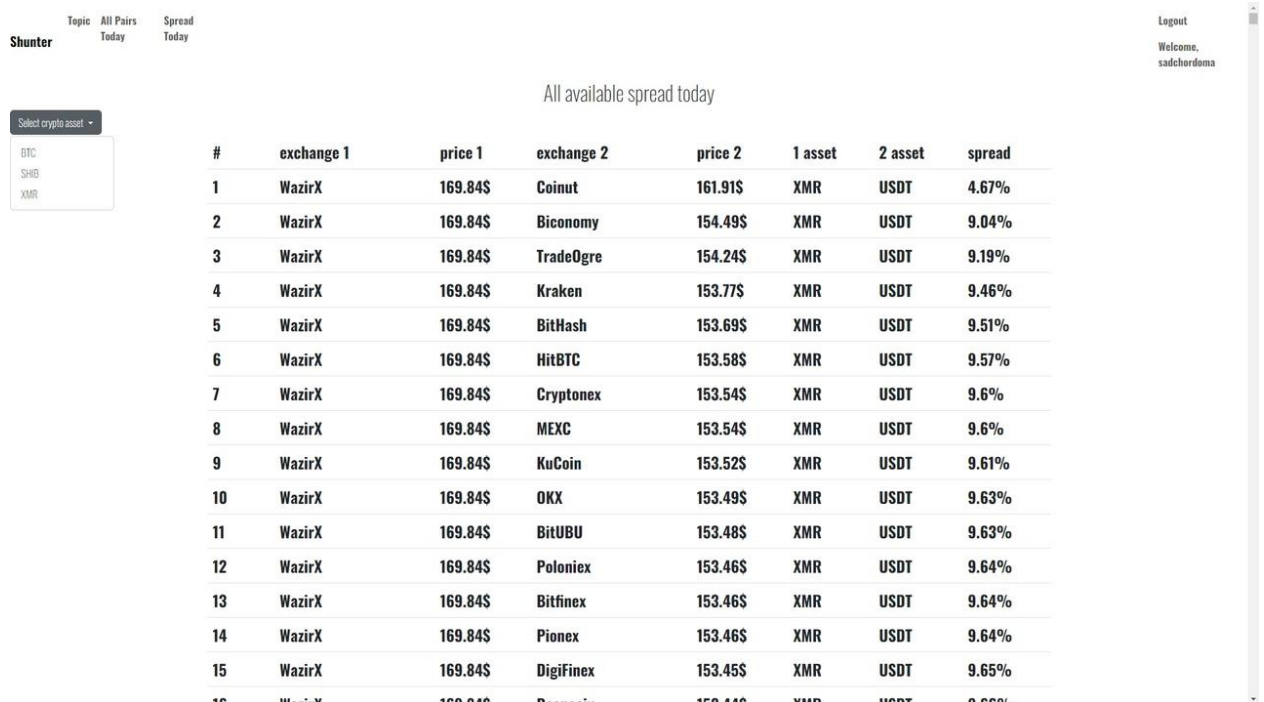
Скриншот основной html – страницы.



The screenshot shows the main page of the Shunter website. At the top left, there are links for 'Shunter', 'Topic', and 'Login'. In the center, there is a 'Top' button and a notification that says 'Press F11 to exit full screen'. Below this is a table with 17 rows of cryptocurrency data.

#	Name
1	bitcoin
2	ethereum
3	tether
4	bnb
5	usd coin
6	xrp
7	cardano
8	dogecoin
9	polygon
10	solana
11	polkadot
12	litecoin
13	binance usd
14	tron
15	shiba inu
16	avalanche
17	dai

Скриншот html – страницы Spread Today. Размещение элементов пользовательского интерфейса: Левый верхний угол – пункты меню, ниже – выпадающий список с доступными активами, справа – приветственное сообщение для пользователя, вошедшего в систему.



The screenshot shows the 'Spread Today' page of the Shunter website. At the top left, there are links for 'Shunter', 'Topic', 'All Pairs', and 'Spread Today'. On the right, there is a 'Logout' button and a welcome message 'Welcome, sashchordoma'. Below the navigation links is a dropdown menu for 'Select crypto asset' with options for BTC, SHIB, and XMR. The main content is a table titled 'All available spread today' with 16 rows of data.

#	exchange 1	price 1	exchange 2	price 2	1 asset	2 asset	spread
1	WazirX	169.84\$	Coinut	161.91\$	XMR	USDT	4.67%
2	WazirX	169.84\$	Biconomy	154.49\$	XMR	USDT	9.04%
3	WazirX	169.84\$	TradeOgre	154.24\$	XMR	USDT	9.19%
4	WazirX	169.84\$	Kraken	153.77\$	XMR	USDT	9.46%
5	WazirX	169.84\$	BitHash	153.69\$	XMR	USDT	9.51%
6	WazirX	169.84\$	HitBTC	153.58\$	XMR	USDT	9.57%
7	WazirX	169.84\$	Cryptonex	153.54\$	XMR	USDT	9.6%
8	WazirX	169.84\$	MEXC	153.54\$	XMR	USDT	9.6%
9	WazirX	169.84\$	KuCoin	153.52\$	XMR	USDT	9.61%
10	WazirX	169.84\$	OKX	153.49\$	XMR	USDT	9.63%
11	WazirX	169.84\$	BitUBU	153.48\$	XMR	USDT	9.63%
12	WazirX	169.84\$	Poloniex	153.46\$	XMR	USDT	9.64%
13	WazirX	169.84\$	Bitfinex	153.46\$	XMR	USDT	9.64%
14	WazirX	169.84\$	Pionex	153.46\$	XMR	USDT	9.64%
15	WazirX	169.84\$	DigiFinex	153.45\$	XMR	USDT	9.65%
16	WazirX	169.84\$	Deepcoin	153.44\$	XMR	USDT	9.66%

Тэг <head> используется для хранения метаданных документа, таких как заголовок страницы, стили, ссылки на скрипты и другую информацию,

которую браузер может использовать для правильного отображения страницы.

Тэг `<nav>` содержит основные функции навигации для страницы. Вторичные ссылки и т.д. не входят в навигацию.

Тэг `<tr>` определяет строку (ряд) таблицы. Этот тэг должен использоваться внутри тэга `<table>`, а строка может содержать ячейки (другие тэги по типу `<th>`, `<td>`).

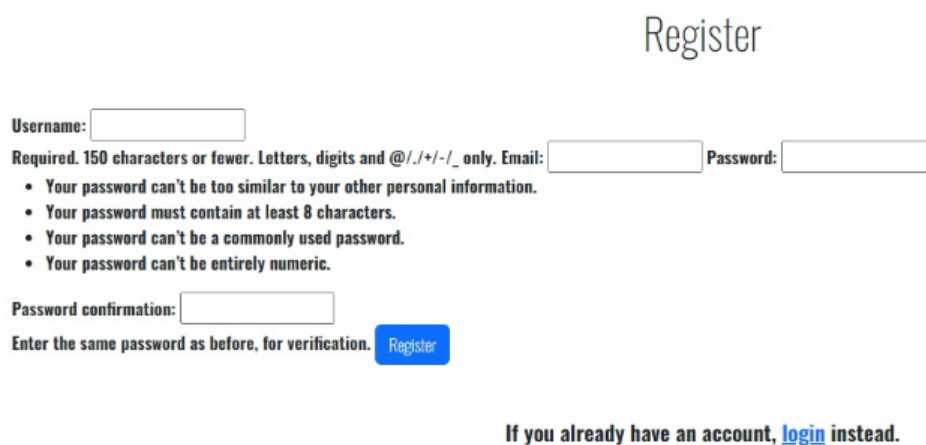
Тэг `<th>` определяет заголовок таблицы и является ячейкой таблицы на пересечении строки заголовка и столбца. В отличие от тэга `<td>`, содержимое тэга `<th>` обычно выравнивается по центру и выделяется жирным шрифтом.

Тэг `<td>` определяет ячейку таблицы внутри строки. Он используется для отображения информации в таблице. Как правило, содержимое тэга `<td>` выравнивается по левому краю.

Тэг `<body>` определяет тело документа, которое отображается в окне браузера. Он содержит всю видимую информацию на странице, такую как текст, изображения, ссылки и т.д.

Скриншоты экранных форм

Регистрация



Register

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. Email: Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

If you already have an account, [login](#) instead.

Вход



Login

Username: Password:

Don't have an account? [Create an account.](#)

Листинг программы

sql- script

```
create table source_files (  
    id integer primary key,  
    filename varchar(255) NOT NULL,  
    processed datetime  
);
```

```
create table crypto (  
    id integer primary key,  
    exchange varchar(255),  
    asset1 varchar(255),  
    asset2 varchar(255),  
    price real,  
    source_file integer NOT NULL,  
    CONSTRAINT fk_source_files  
    FOREIGN KEY (source_file)  
    REFERENCES source_files(id)  
    ON DELETE CASCADE  
);
```

python

ugatu-software-design/webservice/repository/sql_api.py

from typing import List

from .connector import StoreConnector

from pandas import DataFrame, Series

from datetime import datetime

"""

В данном модуле реализуется API (Application Programming Interface) для взаимодействия с БД с помощью объектов-коннекторов.

ВАЖНО! Методы должны быть названы таким образом, чтобы по названию

можно было понять выполняемые действия.

"""

def select_all_from_source_files(connector: StoreConnector) -> List[tuple]:

""" Вывод списка обработанных файлов с сортировкой по дате в порядке убывания (DESCENDING) """

query = f'SELECT * FROM source_files ORDER BY processed DESC'

result = connector.execute(query).fetchall()

return result

def insert_into_source_files(connector: StoreConnector, filename: str):

""" Вставка в таблицу обработанных файлов """

now = datetime.now() # текущая дата и время

date_time = now.strftime("%Y-%m-%d %H:%M:%S") # преобразуем дату в формат SQL, например, '2022-11-15 22:03:16'

query = f'INSERT INTO source_files (filename, processed) VALUES ({filename}, {date_time})'

result = connector.execute(query)

return result

def insert_rows_into_processed_data(connector: StoreConnector, dataframe: DataFrame):

""" Вставка строк из DataFrame в БД с привязкой данных к последнему обработанному файлу (по дате) """

rows = dataframe.to_dict('records')

files_list = select_all_from_source_files(connector) # получаем список обработанных файлов

```

# т.к. строка БД после выполнения SELECT возвращается в виде
объекта tuple, например:
# row = (1, 'seeds_dataset.csv', '2022-11-15 22:03:16'),
# то значение соответствующей колонки можно получить по индексу,
например id = row[0]
last_file_id = files_list[0][0] # получаем индекс последней записи из
таблицы с файлами
if len(files_list) > 0:
    for row in rows:
        connector.execute(f"""
        INSERT INTO crypto (exchange, asset1, asset2, price, source_file)
        VALUES ('{row["Exchange"]}', '{row["asset1"]}', '{row["asset2"]}',
        '{row["price"]}', '{last_file_id}')
        """)
    print('Data was inserted successfully')
else:
    print('File records not found. Data inserting was canceled.')

```

```

def select_rows_from_processed_data(connector: StoreConnector,
source_file: int, asset) -> List[dict]:
    if asset is None:
        return [{}]
    selected_rows = connector.execute(f"""
    SELECT * FROM crypto WHERE source_file = '{source_file}'
    AND asset1 = '{asset}' OR asset2 = '{asset}'
    """)

    dict_pairs = []
    for item in selected_rows.fetchall():
        dict_pairs.append({"id": item[0],
                           "exchange": item[1],
                           "asset1": item[2],
                           "asset2": item[3],
                           "price": item[4],
                           "source_file": item[5]
                           })
    return dict_pairs

```

```

def delete_selected_row(connector: StoreConnector, row: dict):
    connector.execute(f"""
    DELETE FROM crypto WHERE id = {row["id"]};
    """)

```

```
def update_selected_row(connector: StoreConnector, values_to_change:
dict):
    connector.execute(f"""
        UPDATE crypto set exchange = '{values_to_change["exchange"]}',
        asset1 = '{values_to_change["asset1"]}', asset2 =
'{values_to_change["asset2"]}',
        price = '{values_to_change["price"]}', source_file =
'{values_to_change["source_file"]}'
        WHERE id = {values_to_change["id"]};
    """)
```

Выводы к работе.

В ходе выполнения лабораторной работы мы ознакомились с основными методами проектирования БД и методами проектирования пользовательского интерфейса.