МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

БУКОВИНСЬКИЙ ДЕРЖАВНИЙ

ФІНАНСОВО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КОМП’ЮТЕРНИХ ТА СОЦІАЛЬНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

**Пояснювальна записка**

до дипломного проекту освітньо-кваліфікаційного рівня бакалавр

на тему: «**Інформаційна система для аналізу валютних коливань**»

Виконав: студент 4 курсу, групи КН-41

напряму підготовки

6.050101 – комп’ютерні науки

Бурла Єміліан Костянтинович

Керівник:

к.п.н. Яцько Оксана Мирославівна

Рецензент:

к.ф.-.м.н. Спіжавка Дмитро Васильович

Чернівці – 2016

**ЗМІСТ**

Зм.

Аркуш

№ докум.

Підпис

Датаа

Аркуш

6

БДФЕУ.160067.004.ПЗ

Розроб.

Бурла Є.К.

Перевір.

Н. Контр.

*Стеценко І.В.*

Затверд.

*Інформаційна система для аналізу валютних коливань*

Літера

Аркушів

КН-41

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ……………..…………6

ВСТУП……………………………………………………………………………..7

РОЗДІЛ 1. ПРОГРАМНІ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ АНАЛІЗУ КУРСІВ ВАЛЮТ……………………….…………………………….9

* 1. Статистичний аналіз валютних курсів………………………………..9
  2. Огляд аналогів інформаційних систем з курсами валют…………...10
  3. Недоліки і проблеми існуючих розробок…………….………...……12
  4. Постановка задачі для проектування………….……………...……...12

РОЗДІЛ 2. АЛГОРИТМИ ТА МЕТОДИ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ ДЛЯ АНАЛІЗУ ВАЛЮТНИХ КОЛИВАНЬ…..…………...…..14

* 1. Отримання даних з Інтернет за допомогою парсингу HTML….......14
  2. Отримання даних за допомогою API……………………………..….17
  3. Підключення API до свого програмного продукту…………….…...20

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СТВОРЕННЯ ВЕБ-САЙТУ ДЛЯ АНАЛІЗУ ВАЛЮТНИХ КОЛИВАНЬ ………………………………………...23

* 1. Використанні Бібліотеки та фреймворки……………………...…….23
  2. Реалізація та UML-діаграма………………………………………….38

РОЗДІЛ 4. ЗАСТОСУВАННЯ РОЗРОБЛЕННОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПРАКТИЧНИХ ЦІЛЕЙ………………………………..…….42

* 1. Застосування інформаційної системи для курсів валют……….…...42

5. ВИСНОВКИ………………………………………....………………………...48

6. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…………………………………….49

7. ДОДАТКИ……………………………………………………………………..51

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*6*

БДФЕУ.160067.004.ПЗ

⮚ PB – Приват Банк

⮚ NBU – Національний Банк України

⮚ API – Application Programming Interface

⮚ ORM – Object Relational Mapping

⮚ REST – Representational State Transfer

⮚ JSON – JavaScript Object Notation

⮚ AJAX – Asynchronous JavaScript and XML

⮚ HQL – Hibernate Query Language

⮚ UI – User Interface

⮚ POM – Project Object Model

⮚ POJO – Plain Old Java Object

ВСТУП

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*7*

БДФЕУ.160067.004.ПЗ

**Актуальність теми.** На сьогоднішній день розвитку людства та глобалізації економіки багато споживачів цікавляться курсами валют, для знаходження, яких використовують глобальну мережу Інтернет, адже це найпростіший та найзручніший спосіб знайти саме дані, без потреби витрачати час на похід до банківської установи.

Тому метою даного дипломного проекту було створити веб-сайт(проект), за допомогою якого споживачі зможуть переглядати курси валют в зручний для них час з будь-якої точки земної кулі станом на сьогоднішній день або на будь-яку іншу дату, за певний діапазон часу та оформити їх у вигляді таблиці, побудувати графік коливань валют тощо.

**Об’єктом дослідження** є алгоритм і метод створення веб-сайту інформаційного спрямування.

**Предметом дослідження** є структура створення веб-сайту за допомогою певних алгоритмів та API, які надають курси в JSON або XML форматі.

**Мета проекту:** розробити, програмно реалізувати, дослідити, оптимізувати та розмістити веб-сайт для відображення курсів валют в окремих банківських установах України.

Новизна предмету даної розробки полягає в тому, що на фоні розроблених на даний час аналогів, цей продукт матиме можливість працювати з декількома банками, відображати курси валют на діапазони дат, а також відображати курси валют на протязі 3-4 років, графічно відображати коливання курсів валют тощо.

Пояснювальна записка даного дипломного проекту описує дану розробку за наступною структурою:

В першому розділі міститься опис аналогів розробки та формується постановка проблеми для дипломного проекту.

Другий розділ описує методи їх підключення, а також отримання та обробки даних.

Мета третього розділу полягає в розробці інформаційної системи курсів валют в деяких банках України та її тестуванні.

Четвертий розділ описує використання інформаційної системи.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*8*

БДФЕУ.160067.004.ПЗ

Отже, для досягнення мети даного дипломного проекту необхідне вирішення наступних завдань:

* розробка програмного продукту, в якому реалізовано можливість отримання даних з БД для їх обробки;
* можливість отримувати дані в Real Time(-і);
* здійснення аналізу росту ціни покупки та продажу валют;
* візуалізація досліджуваних даних за допомогою графіків.

РОЗДІЛ 1

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*9*

БДФЕУ.160067.004.ПЗ

ПРОГРАМНІ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ АНАЛІЗУ КУРСІВ ВАЛЮТ

**1.1. Статистичний аналіз валютних курсів**

Валютним курсом, називається ціна грошової одиниці однієї держави, виражена в грошовій одиниці іншої. Виділяють кілька видів валютних курсів:

* за способом регулювання: фіксований, плаваючий (формується попитом та пропозицією) і непостійний (встановлюється в певному коридорі);
* за різновидами ринку: поточний, строковий (форварди, ф’ючерси, опціони).

На перший погляд здається, що валютний курс – це звичайний коефіцієнт перерахунку валют, який формується під впливом попиту та пропозиції. Проте основу вартості валютного курсу складає купівельна спроможність валют, яка виражає ціни на товари, послуги та інвестиції. Дана економічна категорія характерна для товарного виробництва і відображає його відношення до світового ринку. Враховуючи, що вартість є великим поняттям для вираження економічних умов виробництва товарів, порівняння національних грошових одиниць різних держав ґрунтується на відношенні вартості, що склалася в ході виробництва та обміну. Валютний курс дозволяє виробникам та покупцям продукції порівнювати національні ціни з цінами в інших країнах. В результаті такого порівняння можна визначити, наскільки вигідно та доцільно розвивати те чи інше виробництво в даній країні. Валютний курс повністю підкоряється дії закону вартості, виражаючи співвідношення національної та світової економіки[1].

Потрапляючи на світовий ринок, національна продукція оцінюється за допомогою інтернаціональної міри вартості. Таким чином, валютний курс сприяє товарному обміну в світовому господарстві. В основі світових цін лежить інтернаціональна ціна виробництва, яка базується на тих цінах, які сформовані в країнах, що постачають свою продукцію на світовий ринок[2].

Процедура формування рівня валютних курсів, називається валютним котируванням. Повне валютне котирування означає, що встановлюється курс покупки і курс продажу[3]:

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*10*

БДФЕУ.160067.004.ПЗ

USD/UAH = 5,4350-5,4850.

Різниця між курсом покупки і курсом продажу, називається курсовим спредом і є джерелом прибутку для суб’єкта, що здійснює валютне котирування. Валютне котирування має такі види:

* одиницям національної валюти відповідає 10, 100 тощо, одиниць іноземною валюти, тобто базою котирування є іноземна валюта: USD/UAH = 5,4350-5,4850 база котирування – USD, валюта котирування – UAH;
* пряме котирування (американський варіант) – визначає, скільком обернене котирування (європейський варіант) – визначає, скільком одиницям іноземної валюти відповідає 10, 100 і т.д. одиниць національної валюти, тобто базою котирування є національна валюта: UAH/USD = 0,1823-0,1840, база котирування – UAH, валюта котирування – USD;

Очевидно, що між рівнями курсів, обчислених за прямим (К) і оберненим котируванням (R), виконується співвідношення[4]:

.

У більшості країн застосовують пряме котирування.

* крос-котирування — визначає курс двох валют одна до одної через курс кожної з них відносно третьої валюти, часто до долара США.
  1. **Огляд аналогів інформаційних систем з курсами валют.**

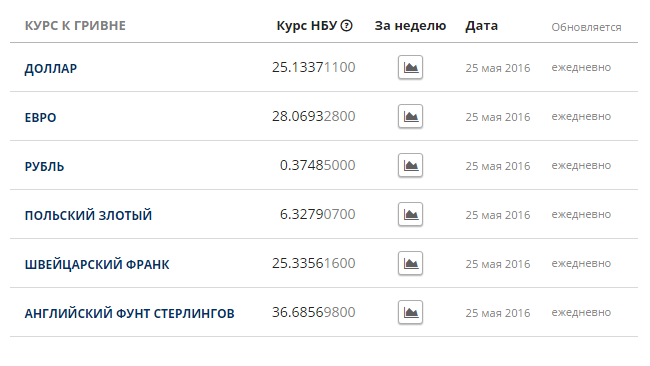
При створенні веб-додатків, веб-сайтів, програмних продуктів тощо, дуже важливим для розробника є аналіз вже існуючих аналогів даної розробки. Це надасть можливість зрозуміти як буде виглядати UI продукту, які можливості та відмінності від аналогів він буде мати.

На самому початку розробник повинен проаналізувати всі «сильні» та «слабкі» сторони розробки, провести тестування та відокремити для себе важливі сторони.

Розробка, даного проекту полягає у написанні веб-сайту, який б дозволяв всім отримувати швидко та точно отримувати курси потрібної валюти на потрібний момент часу, побудувати графік так проаналізувати по ньому яким чином відбувалась зміна вартості потрібної валюти.

Отже, для виконання даного дипломного проекту було проведено пошук аналогів для вищезазначеної програми. Серед ряду існуючих сьогодні аналогів даного програмного продукту було вибрано найбільш популярні, та за функціональними можливостями схожі розроблений в подальшому програмний продукт. Аналогами є наприклад інформери з таких сайтів як «http://privatbank.ua/» та «http://minfin.com.ua/».

Інформер з сайту Мінфіна (рис 1.1) надає інформацію про курси таких валют як: долар, євро, рубль, польський злотий, швейцарський франк та англійський фунт. Також є можливість поспостерігати, як змінювалися курси валют за останній тиждень за допомогою графіка.



*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*11*

БДФЕУ.160067.004.ПЗ

Рисунок 1.1. Інформер курс валют в NBU з сайту «http://minfin.com.ua/»

Інформер з PB (рис. 1.2) підтримує лише базові можливості, надає курси лише на сьогоднішній день у відділеннях Приват Банку та в НБУ.

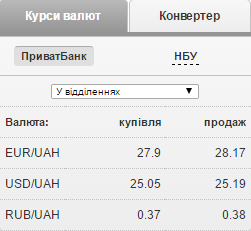


Рисунок 1.2. Інформер курсів валют PB з сайту «http://privatbank.ua»

* 1. **Недоліки і проблеми існуючих розробок**

Найбільшим недоліком існуючих розробок є відсутність вибору конкретної дати, адже можливо користувачу потрібно дізнатися, які були курси валют місяць, рік або більше часу тому для проведення аналізу, написання звітів, доповідей тощо. Другим недоліком є те, що неможливо зробити вибірку курсів на заданий діапазон дат, що може використовуватися для побудови статистичних вибірок тощо. Наступним недоліком є відсутність можливості графічно відобразити за вказаним діапазоном дат коливання курсів валют в інформері від NBU, а також повністю відсутність графіків в інформері від PB. Незручним є те, що для того щоб дізнатися курси в різних банках треба переходити по їх порталам з метою відображення даних.

* 1. **Постановка задачі для проектування.**

Незважаючи на всі сильні сторону порталу Мінфіну і ПриватБанку, вони надають дуже примітивний API, тому користувач зіштовхується з наступними проблемами:

* запит даних з API на діапазон дат;
* обробку отриманих даних в JSON форматі;

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*12*

БДФЕУ.160067.004.ПЗ

* помилки при відправленні запитів до API, а саме: помилки видачі даних з серверів МінФіну та ПриватБанку, помилки зв’язані з відсутністю курсів валют на конкретну дату (може бути спричинено деякими помилками на сервері). Також API МінФіну для курсів з НБУ не видає курси на вихідні дні, тобто запити на суботу, неділю та дні, на яких припадають державні свята (точніше кажучи воно видає пустий масив, без повідомлення про відсутність дат);
* обмежена кількість запитів до API.

Тому для вирішення цих проблем потрібно:

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*13*

БДФЕУ.160067.004.ПЗ

* створити кастомні обробники виключених ситуацій для окремих дат, для дат які припадають на вихідні тощо;
* здійснити перевірку помилок, які виникають на сервері з курсами валют;
* зменшити кількість кіл звернень до API за допомогою перевірки перед кожним колом на сервер, чи є запитувані дані на сервері, в такому випадку вони стягуються з сервера, який розташований на хостингу, тим самим розвантажую кількість кіл обминаючи обмежену кількість дозволених запитів за певний проміжок часу;
* реалізувати візуальне представлення даних у вигляді таблиць і графіків.

РОЗДІЛ 2

АЛГОРИТМИ ТА МЕТОДИ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ ДЛЯ АНАЛІЗУ ВАЛЮТНИХ КОЛИВАНЬ

* 1. **Отримання даних з мережі Інтернет за допомогою парсингу HTML**

Web Mining – процес вибірки даних з веб-ресурсів, який, як правило, має більш практичну ніж теоретичну значущість. Основна ціль – збір даних (parsing) з подальшим збереженням в потрібному форматі. Фактично, задача зводиться до написання HTML парсерів.

Існує декілька підходів до вилучення даних:

1. Аналіз DOM-дерева, використання XPath.
2. Парсінг рядків.
3. Використання регулярних виразів.
4. XML парсінг.
5. Візуальний підхід.

*Аналіз DOM-дерева*

Цей підхід ґрунтується на аналізі DOM-дерева при використанні, якого дані можна отримати напряму по ідентифікатору, імені або інших атрибутів елемента дерева (таким елементом може служити параграф, таблиця, блок тощо). Крім того, якщо елемент не позначений ідентифікатором – до нього можна дістатися по унікальному шляху, спускаючись вниз по DOM-дереву. Наприклад:

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*14*

БДФЕУ.160067.004.ПЗ

*body -> p[10] -> a[1] -> текст посилання.*

Або пройти по списку однотипних елементів, наприклад:

*body -> links -> 5 элемент -> текст посилання*

Перевагами цього підходу є:

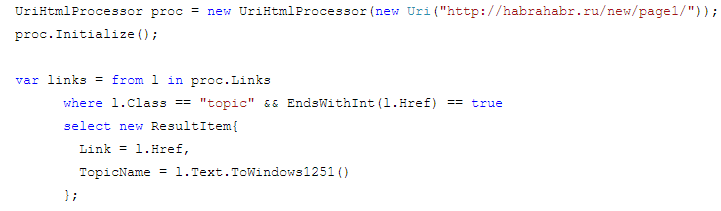
* можна отримати дані будь-якого типу і будь-якого рівня складності;
* знаючи, де знаходиться елемент, можна отримати його значення, прописавши шлях до нього.

Недоліки цього підходу:

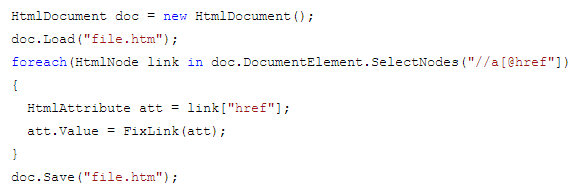
* різні HTML/JavaScript Engines по різному генерують DOM дерево, тому треба прив’язатись до конкретної системи.
* шлях елемента може змінюватися, тому, як правило, такі парсери розраховані на короткий період збору даних;
* DOM-шлях може бути складний і не завжди однозначний.

Цей підхід можна використовувати разом з бібліотекою Microsoft.mshtml, яка, по суті являється core елементом в Internet Explorer[7].

Data Extracting SDK використовує Microsoft.mshtml для аналізу DOM дерева, але є «надстройкою» над бібліотекою для зручності роботи:



Наступним етапом еволюції аналізу DOM-дерева є використання XPath – тобто, шляхів, які широко використовують при парсінгу XML-даних. Суть даного підходу в тому, що з допомогою якогось простого синтаксису описати шлях до елементу без необхідності поступово рухатись вниз по DOM-дереву. Даний підхід використовує всім відома бібліотека jQuery та бібліотека HtmlAgilityPack:



*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*15*

БДФЕУ.160067.004.ПЗ

*Парсинг рядків.* Не дивлячись на те, що цей підхід не можна застосовувати для написання важливих парсерів, іноді дані відображаються за допомогою деякого шаблону, тоді значення параметрів стандартні, а змінюються тільки їх значення. В такому випадку дані можуть бути отримані без аналізу DOM-дерева, а шляхом парсингу рядків, наприклад як це зроблено в Data Exctracting SDK:

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*16*

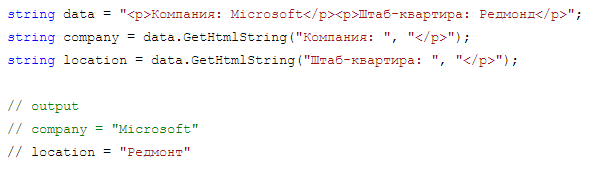
БДФЕУ.160067.004.ПЗ

Дані:

Компанія: Microsoft

Штаб-квартира: Редмунд

Код:



Використання набору методів для аналізу рядків іноді більш ефективний аніж аналіз DOM-дерева або XPath.

Регулярні вирази і парсинг XML. Дуже зустрічається, коли HTML повністю парсили з допомогою регулярних виразів. Це невірний підхід, так як таким чином можна отримати більше проблем, ніж користі.

Регулярні вирази необхідно використовувати тільки для вилучення даних, які мають строгий формат – електронні адреси, телефони тощо, в рідких випадках – адреси, шаблонні дані.

Ще одним неефективним підходом є розглядати HTML як XML-дані. Причина в тому, що HTML рідко буває валідним, тобто, таким, що його можна розглядати як XML-дані. Бібліотеки, які реалізують таким підходом більше часу виділяють перетворенню HTML в XML, і вже потім безпосередньому парсингу даних. Тому краще уникати цей підхід[8].

*Візуальний підхід*. В даний момент, візуальний підхід знаходиться на початковій стадії розвитку. Суть підходу в тому, що користувач зміг без використання програмної мови чи API налаштувати систему для отримання потрібних даних будь-якої складності. Вважаю, що парсери майбутнього будуть візуальними.

Проблеми при парсингу HTML даних – використання JavaScript/AJAX/ асинхронних завантажень дуже ускладнюють написання парсерів; різні системи для рендеру HTML можуть видавати різні DOM-дерева; великі об’єми даних потребують писати розподіленні парсери, що призводить до додаткових затрат на синхронізацію.

Не можна одночасно виділити підхід, який буде 100% застосовуватись у всіх випадках, тому сучасні бібліотеки для парсинга HTML даних, як правило, комбінують, різні підходи.

* 1. **Отримання даних за допомогою API**

Як вже будо сказано, API (Application Programming Interface, або інтерфейс для програмування додатків) – це, в першу чергу, інтерфейс. Інтерфейс який дозволяє розробникам використовувати готові блоки для побудови додатку. В випадку з розробкою мобільних додатків, в ролі API може виступати бібліотека для роботи з «розумним будинком» – всі нюанси реалізовані в бібліотеці, і Ви лиш звертаєтесь до цього API в своєму коді.

У випадку розробки веб-додатків, API може видавати дані в іншому від стандартного HTML форматі, завдяки цьому їм буде зручно користуватися при написанні власних додатків. Сторонні загальнодоступні API частіше всього видають дані в одному з двох форматів: JSON або XML. На випадок, якщо розробник ще не вирішив зробити API для свого додатку, йому треба запам’ятати, що JSON набагато більш лаконічний і простий в прочитанні, ніж XML, а сервіси, які надають доступ до даних в XML-форматі, поступово зникають.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*17*

БДФЕУ.160067.004.ПЗ

Проаналізуємо API веб-додатків на прикладах. Деякі додатки – наприклад, Github – мають власний API, яким можуть скористатися інші розробники. То, як вони будуть користуватися ним залежить від можливостей, які надає API і від того, наскільки добре працює фантазія в розробників. API Гітхаба дозволяє, наприклад, отримувати інформацію про користувача, його аватарі, підписників, репозиторіїв і багато іншої корисної і цікавої інформації.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*18*

БДФЕУ.160067.004.ПЗ

Якщо взяти, до прикладу, API Твітера, то інтерфейс цього сервісу може надавати всю інформацію про твіти користувача, його читачів і тих, кого він читає тощо. Це лиш маленька частина всіх можливостей, якими кожний бажаючий може відтворити, використовуючи API стороннього сервісу або створюючи свій власний.

На основі API будуються такі проекти, як карти 2GIS, мобільні додатки, десктопні клієнти для Twitter і Vkontakte. Всі їх функції стали можливими лиш тому, що власні сервіси мають якісні та детально документовані API.

Стандартний запит від стороннього API виглядає приблизно так:



Рисунок 2.1. Стандартний запит від стороннього АРІ

На випадок, якщо хтось не знає, зазначу, що curl не має жодного відношення до API і використовується в операційних системах для відправки і отримання даних через термінал.

Подібним образом можна надсилати запит на будь-якій мові. Відповіддю на такий запит будуть дані відображені на рис.2.2.

Як видно з цього блоку, відповідь містить в собі логін, аватар, посилання на профіль на сайті і в API, статус користувача, кількість публічних репозиторіїв та іншу корисну і цікаву інформацію.

Існує декілька ситуацій, в яких розробник може захотіти створити API для власного додатку.

1. Мобільний додаток. Багато мобільних додатків для різних сервісів працюють при використанні API-сервісів. Розробник описує API, розробляє простий мобільний додаток і клієнт зі смартфоном буде отримувати інформацію в своє задоволення саме через API. Це зручно і має сенс[9].



Рисунок 2.2. Відповідь на запит

1. OpenSource. Все стає краще, якщо використовувати OpenSource. Насправді якщо склалася ситуація, коли в додатку з’явилася аудиторія, яка користується ним, чому б не обернути все собі ж на користь та на користь самої аудиторії, звичайно. Треба створити своє API, за допомогою якого користувачі даного додатку при бажанні зможуть створити нові клієнти для нього, нові сервіси на його основі, і можливо, розкриють його нові можливості.
2. Максимальне розділення фронтенда і бекенда. Наприклад, при використовуванні фронтенд-фреймворків.

Одного API недостатньо. Створити повноцінний API для свого додатку – половина всієї роботи. Щоб звернутися до API, в першу чергу потрібно відправити HTTP-запити з метою отримання потрібну інформації. Значно розумніше буде створити бібліотеку для роботи з інтерфейсом, в якій будуть описані всі необхідні способи отримання і відправки інформації за допомогою API.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*19*

БДФЕУ.160067.004.ПЗ

Ще раз використовуємо Github для приведення прикладу: для роботи з API цього сервісу створено декілька бібліотек на різних мовах, наприклад gem Octokit. В документації до таких бібліотек кожний зацікавлений розробник зможе знайти всі необхідні способи отримання інформації з Гітхабу і її зворотної відправки через API-сервіси.

Таким чином, якщо розробник створює власний API, йому треба задуматись, можливо йому треба створити такі ж бібліотеки для роботи з додатком на найбільш розповсюджених мовах. І треба бути готовим, що хтось може створити власну бібліотеку для роботи з API, який створив розробник додатку.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*20*

БДФЕУ.160067.004.ПЗ

* 1. **Підключення API до свого програмного продукту**

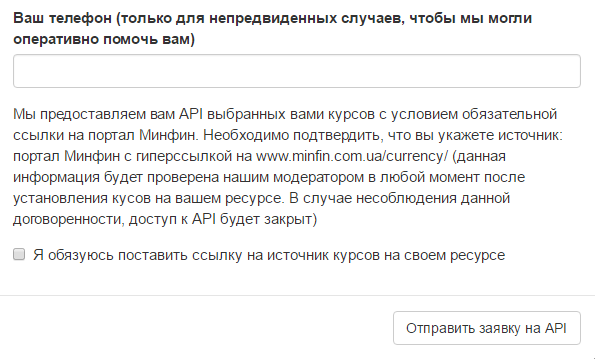
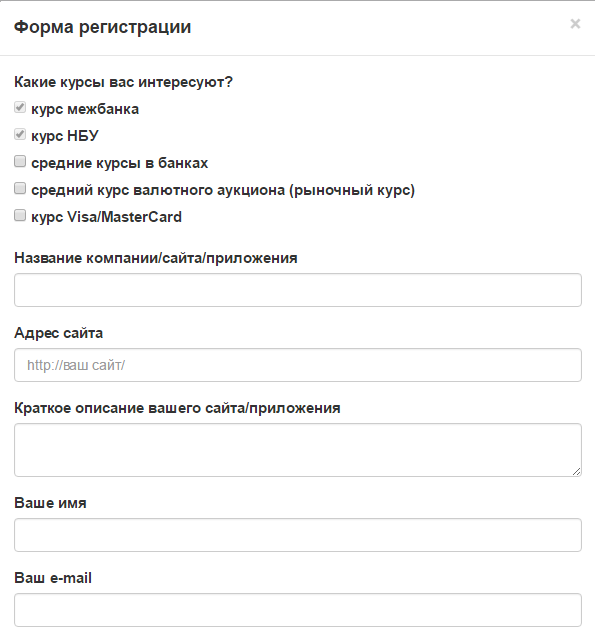


Рисунок 2.3. Форма реєстрації для отримання точена

Для підключення API МінФіну потрібно:

* надати МінФіну свою назву та адресу свого сайту для формування ключа доступу (токена), заповнюючи форму (рис.2.3);
* отримати токен (40-а символьний) по повідомленню на свій email;
* налаштувати систему отримання даних;
* врахувати, що кількість запитів не повинно перевищувати 1 запит раз в 5 хвилин;
* вказати посилання на Фінансовий портал «МінФін» на своєму сайті в місці, де будуть показуватись курси валют.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*21*

БДФЕУ.160067.004.ПЗ

Результат запиту інформації с API:

*{"usd":{"date":"2015-06-25 00:00:00", "ask":"21.17802000", "bid":"21.17802000", "trendAsk":"-0.34770000","trendBid":"-0.34770000", "currency":"usd"}, "eur":{"date":"2015-06-25 00:00:00", "ask":"23.74691400", "bid":"23.74691400", "trendAsk":"-0.37050300", "trendBid":"-0.37050300", "currency":"eur"} ,"rub":{"date":"2015-06-25 00:00:00", "ask":"0.39164000", "bid":"0.39164000", "trendAsk":"-0.00545000", "trendBid":"-0.00545000", "currency":"rub"}}*,

де:

* *Ключ* – валюта;
* *date* – дата;
* *bid* – продаж;
* *ask* – купівля;
* *trendBid* - тренд продажу;
* *trendAsk* - тренд купівлі.

Для отримання даних з minfin.com.ua необхідно надіслати параметр заголовка user-agent. Формат user-agent може бути таким:

*"[ApplicationName]/[Version] ([http://link-to-your-application.com])"*,

де:

* ApplicationName – назва додатку;
* Version – версія;
* *http://link-to-your-application.com –* посилання на додаток/сайт, чи коротке описання. Наприклад, "FinApplicationBot/1.0 (http://test.com)";
* встановити user-agent в Java можна таким чином: System.setProperty("http.agent", "").

Рекомендується не використовувати user-agent браузерів(крім тих випадків, коли запит виконується з-під браузера, наприклад за допомогою JS).

РОЗДІЛ 3

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*22*

БДФЕУ.160067.004.ПЗ

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*23*

БДФЕУ.160067.004.ПЗ

ПРОГРАМНА РЕАЛІЗАЦІЯ СТВОРЕННЯ ВЕБ-САЙТУ ДЛЯ АНАЛІЗУ ВАЛЮТНИХ КОЛИВАНЬ

* 1. **Використанні Бібліотеки та фреймворки**

Бібліотека (від англ. library) – збірка об’єктів або підпрограм для вирішення близьких за тематикою задач. У залежності від мови програмування бібліотеки містять об’єктні модулі або сирцевий (англ. source code; також перекладається українською як вихідний код, програмний код, джерельний код, первинний код, текст програми, у професійному середовищі також сирцевий код, у контексті код або сирці) код та дані, допоміжні для використання та інтеграції нових можливостей в програмні рішення. Бібліотека може означати те саме, що модуль, або декілька модулів.

З точки зору комп’ютерних наук бібліотеки діляться на статичні та динамічні:

* **Статичні бібліотеки**. Можуть бути у вигляді початкового тексту, що підключається програмістом до своєї програми на етапі написання, або у вигляді об’єктних файлів, що приєднуються (лінкуються) до виконуваної програми на етапі компіляції (у Microsoft Windows такі файли мають розширення .lib, у UNIX-подібніх ОС – зазвичай .a). В результаті програма включає всі необхідні функції, що робить її автономною, але збільшує розмір.
* **Динамічні бібліотеки**. Також називаються розподілюваними бібліотеками (англ. shared library), або бібліотеками, що динамічно підключаються (англ. Dynamic Link Library, DLL). Це окремі файли, що надають програмі набір використовуваних функцій для завантажування на етапі виконання при зверненні програми до ОС із заявкою на виконання функції з бібліотеки. Якщо необхідна бібліотека вже завантажена в оперативну пам’ять, програма використовуватиме завантажену копію бібліотеки. Такий підхід дозволяє зекономити час та пам’ять, оскільки декілька програм використовують одну копію бібліотеки, вже завантажену в пам’ять. Динамічні бібліотеки зберігаються зазвичай у визначеному місці й мають стандартне розширення[15]. При написанні програми програмісту досить вказати транслятору мови програмування (компілятору або інтерпретатору), що слід підключити таку-от бібліотеку і використовувати таку-от функцію з вказаної бібліотеки. Ні початковий текст, ні виконуваний код функції до складу програми не входить.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*24*

БДФЕУ.160067.004.ПЗ

Для автоматизації зборки проекту використовувався Maven. Maven – це засіб автоматизації роботи з програмними проектами, який використовується для управління (management) та складання (build) програм.

Вся структура проекту описується в файлі pom.xml (POM – Project Object Model, рис 3.1.), який повинен знаходитись в кореневій теці проекту. Ключовим поняттям Maven являється артефакт – це, по суті, будь-яка бібліотека, збережена в репозиторії. Це може бути будь-яка залежність(dependency) або плагін.

Dependencies – це ті бібліотеки, які безпосередньо використовуються в проекті для компіляції коду або його тестування.

Плагіни – використовуються Maven-м при складанні проекту чи для інших цілей (деплоймент, створення файлів проекту для Eclipse тощо).

Для опису програмного проекту, який потрібно побудувати (build), Maven використовує конструкцію відому як Project Object Model (POM), залежності від зовнішніх модулів, компонентів та порядку побудови. Виконання певних, чітко визначених задач – таких, як компіляція коду та пакетування відбувається шляхом досягнення заздалегідь визначених цілей (targets). Ключовою особливістю Maven є його мережева готовність (network-ready).

Двигун ядра може динамічно завантажувати плагіни з репозиторію, того самого репозиторію, що забезпечує доступ до багатьох версій різних Java-проектів з відкритим кодом, від Apache та інших організацій та окремих розробників. Цей репозиторій та його реорганізований наступник – Maven 2 репозиторій, намагається бути де-факто механізмом для дистрибуції Java програм, але прийняття його в такій якості йде повільно.

Maven забезпечує підтримку побудови не просто перебираючи файли з цього репозиторію, але й завантажуючи назад артефакти у кінці побудови. Локальний кеш звантажених артефактів діє, як первісний засіб синхронізації виходу проектів на локальній системі.

Maven базується на плаґін-архітектурі, що дозволяє зробити використання будь-якої програми контрольованим через стандартний вхід. Теоретично, це могло б дозволити будь-кому писати плаґіни для інтерфейсу з інструментами для побудови (компілятори, тестери тощо) для будь-якої мови. В дійсності, підтримка і використання для мов відмінних від Java були мінімальною.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*25*

БДФЕУ.160067.004.ПЗ

Життєвий цикл мавена доволі очікуваний:

* validate – перевіряє коректність метаінформації проекту;
* compile – компілює сирцевий код;
* test – проганяє тести класів з попереднього кроку;
* package – запаковує скомпільовані класи в зручно переміщуваний формат (jar або war, наприклад);
* integration-test – відправляє запаковані класи в середу інтеграційного тестування і проганяє тести;
* verify – провіряє коректність пакету і задоволені вимоги якості;
* install – заганяє пакет в локальний репозиторій, звідки він(пакет) буде доступний для використання як залежність в інших пакетах;
* deploy – відправляє пакет на віддалений production сервер, звідки інші розробники можуть його отримати й використати.



Рисунок 3.1. Файл конфігурації pom.xml з проекту.

Підтягнуті за допомогою Maven-а бібліотеки, які були використанні в проекті:

**Java EE API** – включає в себе декілька технологій, які розширяють деякі функціональні можливості Java SE API-інтерфейсів. Наприклад, включає в себе: «javax.servlet.\*», «javax.websocket.\*», «javax.faces.\*», «javax.faces.component.\*», «javax.enterprise.inject.\*», «javax.validation.\*», «javax.persistence.\*» та інші[5].

**Java Mail API** – надає платформо-незалежні та протокол-незалежні фреймворки для mail та messaging додатків.

**Hibernate ORM Framework.** ORM (Object Relational Mapping) рішенням для Java, є технологія Hibernate, яка не тільки піклується про зв’язок Java класів з таблицями бази даних (і типів даних Java в типи даних SQL), але й надає засоби для автоматичної побудови запитів і вилучення даних, що може значно зменшити час розробки, який зазвичай витрачається на ручне написання SQL та JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки набору даних і конвертації об’єктів, зберігаючи додаток синхронізуючим зі всіма SQL базами даних.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*26*

БДФЕУ.160067.004.ПЗ

Hibernate є вільним програмним забезпеченням, яке поширюється на умовах GNU Lesser General Public License. Hibernate надає легкий для використання каркас (фреймворк) для відображення між об’єктно-орієнтованою моделлю даних і традиційною реляційною базою даних.

Hibernate забезпечує прозору підтримку збереження даних, тобто їхньої персистентності (англ. persistence) для «POJO»-об’єктів, тобто для звичайних Java-об'єктів; єдина сувора вимога до класу, що зберігається – конструктор за замовчуванням (для коректної поведінки у деяких застосуваннях потрібно приділити особливу увагу до методів equals() і hashCode()).

Mapping (зіставлення, буквально – картування) Java класів з таблицями бази даних здійснюється за допомогою конфігураційних XML-файлів або Java-анотацій. При використанні файлу XML, Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання (persistent). У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

Забезпечуються можливості з організації відношення між класами «один-до-багатьох» і «багато-до-багатьох». На додаток до управління зв’язками між об’єктами, Hibernate також може керувати рефлексивними асоціаціями, де об’єкт має зв’язок «один-до-багатьох» з іншими примірниками свого власного типу даних[6].

Hibernate підтримує відображення користувацьких типів значень. Це робить можливим такі сценарії:

* перевизначення типу за замовчуванням SQL, який Hibernate вибирає при відображенні стовпчика властивості.
* картування перераховуваного типу Java до колонок БД, так ніби вони є звичайними властивостями.
* Картування однієї властивості в декілька колонок.

Hibernate забезпечує прозоре збереження POJO (Plain Old Java Objects – простих старих об'єктів Java). Єдина сувора вимога для персистентного класу – конструктор без аргументів, не обов’язково публічний. Для правильної поведінки деяких програм також потрібна особлива увага до методів equals() і hashCode().

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*27*

БДФЕУ.160067.004.ПЗ

Колекції об’єктів даних, як правило, зберігаються у вигляді колекцій Java-об’єктів, таких як набір (Set) і список (List). Підтримуються узагальнені класи (Generics), введені в Java 5. Hibernate може бути налаштований на «ледачі» (відкладені) завантаження колекцій. Відкладені завантаження є варіантом за замовчуванням, починаючи з Hibernate 3.

Зв’язані об'єкти можуть бути налаштовані на каскадні операції. Наприклад, батьківський клас, Album (музичнй альбом), може бути налаштований на каскадне збереження і/або видалення свого нащадка Track. Це може скоротити час розробки і забезпечити цілісність. Функція перевірки зміни даних (dirty checking) дозволяє уникнути непотрібного запису дій в базу даних, виконуючи SQL оновлення тільки при зміні полів персистентних об'єктів.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*280*

БДФЕУ.160067.004.ПЗ

Hibernate забезпечує використання SQL-подібної мови Hibernate Query Language (HQL), яка дозволяє виконувати SQL-подібні запити, записані поряд з об'єктами даних Hibernate. Запити критеріїв надаються як об'єктно-орієнтована альтернатива до HQL[10].

В проекті всі запити до БД реалізуються за допомогою інтерфейсу org.hibernate.Criteria.

Інтерфейс org.hibernate.Criteria являє собою об’єктно-орієнтований запит на вибірку в відношенні до конкретної сутності(Entity) і дозволяє виконувати запити до БД без написання SQL коду. Використання Criteria являється найбільш вдалим підходом для інтерфесів пошуку зі зміною кількістю умов. Для створення екземплярів Criteria використовується клас Session, який виступає в ролі фабрики. Приклад створення і обробки Criteria з проекту:

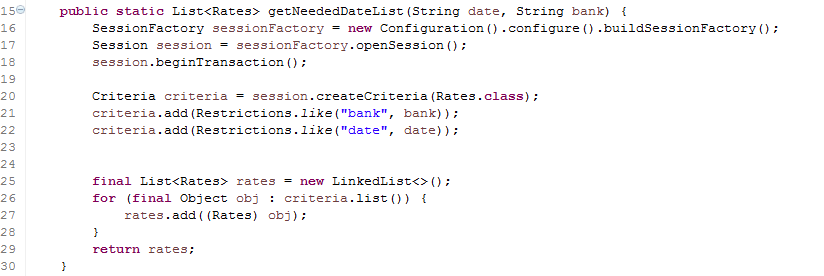


Рис 3.2. Обробка запитів за допомогою Criteria

В результаті виконання даного коду буде здійсненна вибірка даних з урахуванням умов, таких як назва банку і дата яка приходить з запиту який надсилається з клієнтської частини.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*29*

БДФЕУ.160067.004.ПЗ

Умови задаються за допомогою класу Restrinctions. В нього включені такі методи як:

* like(запит по входженню строки в підстроку;
* between(запит по діапазону);
* ilike(case-insensetive запит по входженню строки в підстроку);
* isNotEmpty(вибірка по не пустим елементам) та багато інших;
* та багато інших.

Також динамічна вибірка setFetchMode():

List cats = sess.createCriteria(Cat.class)

.add( Expression.like("name", "Fritz%") )

.setFetchMode("mate", FetchMode.EAGER)

.setFetchMode("kittens", FetchMode.EAGER)

.list();

Цей запит буде отримувати mate та kittens використовуючи outer join.

Hibernate може використовуватись як у самостійних програмах Java, так і в програмах Java EE, що виконуються на сервері (наприклад, сервлети чи EJB session beans). Також він може включатись як додаткова можливість до інших мов програмування. Наприклад, Adobe інтегрував Hibernate у дев'яту версію ColdFusion (що запускається на серверах з підтримкою додатків J2EE) з рівнем абстракції нових функцій і синтаксису, доданих до CFML[10].

Таблиця з БД позначається в Java класі з допомогою Hibernate анотацією @Entity. Поля можна оголошувати як просто «private String bank;» яке буде записуватись в БД з такою ж назвою як оголошено в змінній, так і за допомогою анотації @Column з атрибутами name, type, not null, length, precision та іншими, наприклад:

@Column (name = ”BankName”, length = ”50”, type = ”String”)

private String bank;

Поля які не треба заносити з ентіті класу в БД позначаються анотацією @Transient.

В ентіті класі обов’язково повинно бути поле типу int чи long з анотацією @Id, яка буде відображати Id запису в самій БД. Наприклад для того що би Id заповнявся автоматично з добавленням нового запису треба написати такий код:

@Id @GeneratedValue(GenerationType.AUTO)

private long id;

Код файлу конфігурації Hibernate для підключення бази даних:

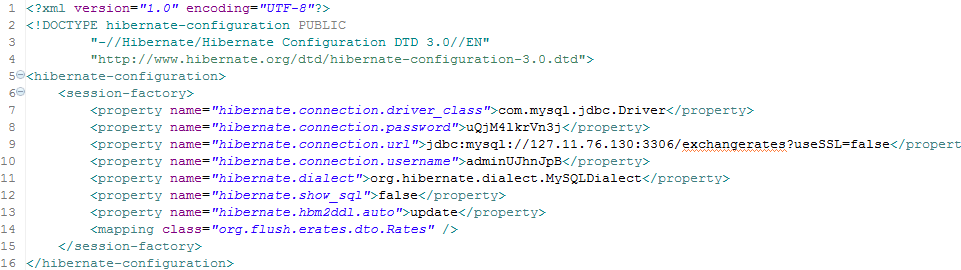


Рисунок 3.3. Підключення бд за допомогою файлу конфігурації hibernate.cfg.xml

**JSON** – (англ. JavaScript Object Notation, укр. об’єктний запис JavaScript, вимовляється джейсон) – це текстовий формат обміну даними між комп’ютерами. JSON базується на тексті, і може бути з легкістю прочитаним людиною. Формат дозволяє описувати об’єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією). Розробив і популяризував формат Дуглас Крокфорд.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*30*

БДФЕУ.160067.004.ПЗ

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовуєтсья в AJAJ, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером[11]. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об’єкти.

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Якщо говорити про веб-застосунки, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси). Формат JSON так само добре підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші.

Приклад використання JSON:

**var** ajaxData = '{"name": "wiki", "fname": "pedia", "rates": [1, 4, 5, 6]}';

**var** ajaxObj = JSON.parse( ajaxData );

alert (ajaxObj.name + ajaxObj.rates[2]);

JSON будується на двох структурах:

* набір пар ім'я/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом;
* впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*31*

БДФЕУ.160067.004.ПЗ

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі їхні форми:

* об'єкт – це невпорядкована множина пар ім’я/значення. Об'єкт починається з символу { і закінчується символом }. Кожне значення слідує за : і пари ім’я/значення відділяються комами;
* масив – це впорядкована множина значень. Масив починається символом [ і закінчується символом ]. Значення відділяються комами;

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*32*

БДФЕУ.160067.004.ПЗ

* значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об’єктом, або масивом. Ці структури можуть бути вкладені одна в одну;
* рядок – це впорядкована множина з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються із зворотної косої риски (backslash). Символи представляються простим рядком.

Тип Рядок (String) дуже схожий на String в мовах C і Java. Число теж дуже схоже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

Наступний приклад показує JSON представлення об'єкта, що описує людину. У об’єкті є рядкові поля імені і прізвища, об'єкт, що описує адресу, і масив, що містить список телефонів.

{ "firstName": "Іван",

"lastName": "Коваленко",

"address": {

"streetAddress": "вул. Грушевського 14, кв.101",

"city": "Київ",

"postalCode": 21000

},

"phoneNumbers": [

"044 123-1234",

"050 123-4567"

]

}

Наступний фрагмент коду JavaScript, який написанний в проекті показує, як клієнт може використати XMLHttpRequest для запиту об’єктів у форматі JSON з серверу:

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*33*

БДФЕУ.160067.004.ПЗ

$.ajax({

url : '/rest/currencies/' + path,

method : method,

data : dataSendObj,

dataType : 'json'

});

**REST** (Representational State Transfer) **–** це архітектурний підхід в дизайні розподілених додатків, наприклад, веб-сервісів. Основними цілями використання цього підходу являються: розширюваність (в плані введення нових компонентів), уніфікований інтерфейс доступу, незалежне розгортання компонентів.

REST-архітектура має декілька обмежень. По-перше, дана архітектура описує взаємодію клієнта і сервера, відповідно може використовуватись не тільки в випадках, в яких є клієнт і сервер. По-друге, вона не має на увазі залежності від користувацького контексту. По-третє, уніфікований інтерфейс, не дозволяє надсилати довільні запити. REST-архітектура побудована на основі протоколу HTTP, використовуючи для відправки запитів цього протоколу: GET, POST, PUT, DELETE, для визначення дії, яку необхідно виконати з вхідними параметрами.

Основним поняттям, на якому будується REST, являється поняття ресурсу (деякого джерела даних), який задається унікальним ідентифікатором (наприклад, в HTTP це URI). При маніпуляції з ресурсами використовується деякий стандартизований інтерфейс доступу.

Ресурси можуть містити підресурси (дочірні ресурси), які містять інформацію про деяку частину ресурсу[12]. Одним з найбільш розповсюджених застосувань REST являються RESTfull- веб сервіси.

**RESTfull Web Services (веб API) –** це веб-сервіс, побудований на основі HTTP і принципів REST. Це набір ресурсів, які характеризуються наступними ознаками:

* базовий URI веб-сервісу, наприклад «http://yoursite.yourdomain/resources/».
* підтриманими типами даних.
* набором операцій підтриманими сервісом, відповідним HTTP-методам (GET, POST, PUT, DELETE та інші).
* API повинен бути основаним на гіпертексті (тобто HTML).

Таким чином працюють різні HTTP-методи для ресурсів і під ресурсів.

Батьківський ресурс: «http://yoursite.yourdomain/resources/resource1/»:

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*34*

БДФЕУ.160067.004.ПЗ

* GET: вертає список дочірніх ресурсів, і можливо деякі інші їх особливості;
* POST: створює новий дочірній елемент колекції resource1;
* PUT: повністю заміщає ресурс resource1;
* DELETE: видаляє ресурс resource1.

Дочірній ресурс: «http://yoursite.yourdomain/resources/resource1/ element1»:

* GET: вертає детальний опис дочірнього ресурсу зі всіма його особливостями;
* POST: часто не використовується для листових вузлів. Працює з підресурсом element1, як з колекцією, добавляє до нього дочірній ресурс;
* PUT: заміщає вказаний дочірній ресурс, а в випадку, якщо його не існує – створює;
* DELETE: видаляє ресурс element1.

Для RESTfull веб-сервісу, основними даними являються URI ресурсу на сервері, і який HTTP метод буде використовуватися. JAX-RS визначає анотації за допомогою котрих можна отримувати цю інформацію. Кожен клас, який являється ресурсом, потрібен мати, хоча б, одну з цих анотацій.

Анотація @Path вказує URI ресурсу. Данна анотація була об’явлена в javax.ws.rs.Path, і може використовуватись для того, щоб змінювати класи і методи. Вона приймає один параметр: строку, яка являється шаблоном URI ресурсу. Шаблон URI вказує місцезнаходження ресурсу. Як показано в прикладі, такий шаблон може містити наступні компоненти: необроблені складові шляху і ідентифікатори параметрів, заключенні в фігурні скобки.

**JAX-RS –** Java API для RESTfull Web Services. Це Java запрограмована мова API, яка підтримує створення веб сервісів (web services) згідно з REST (Representational State Transfer) архітектурним патерном. JAX-RS використовує анотації, впроваджені ще в Java SE 5, для спростування деплойменту веб сервісів для клієнтів та ендпоінтів (endpoints).

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*35*

БДФЕУ.160067.004.ПЗ

З версії 1.1, JAX-RS є офіційно частиною Java EE 6. Примітною особливістю є те, що для початку роботи з JAX-RS не потрібно проводити конфігурацію. Для середовищ відмінних від Java EE 6 потрібно провести конфігурацію в файлі web.xml (дескриптор депройменту).

@Path("/resources/{resource}/{element}")

Наведенному шаблону буде, наприклад, відповідати URI /resources/database/users. Котрий буде означати, запит к таблиці users з БД. Таким чином, значения resource буде рівне database, а значення element рівне users. В залежності від того, що позначається анотацією, шаблон буде застосовуватися к повному шляху, або к шляху відносно батьківського ресурсу. Дане правило ілюструє наступний приклад: package demo.jaxrs:

import javax.ws.rs.Path;

@Path("/colorservice/")

public class ColorService {

public ColorService(){}

@GET

@Path("/{colorCode }/")

public Color getColor(@PathParam("colorCode") String code) {

}

В JAX-RS є 5 анотацій, які відносяться до HTTP методів: javax.ws.rs.DELETE, javax.ws.rs.GET, javax.ws.rs.POST, javax.ws.rs.PUT, javax.ws.rs.HEAD. Для того, щоб вказати, що даний метод класу повинен обробляти запити відправленні деяким HTTP методом, необхідно помітити даний метод класу анотацією, наприклад @GET або @DELETE. Кореневий ресурсний клас, являється точкою входу в RESTful веб-сервісу на JAX-RS. Він позначений анотацією @Path, яка вказує URI кореневого ресурсу сервісу. Його методи або напряму реалізують операції з ресурсом, або представляють доступ до підресурсів.

Вимоги до кореневого ресурсного класу:

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*36*

БДФЕУ.160067.004.ПЗ

* даний клас повинен позначатися анотацією @Path. Вказаний шлях являє собою кореневий URI для всіх ресурсів, які обробляються даним сервісом. Якщо клас кореневого ресурсу вказує на те що його шлях resource і один з його методів позначений анотацією GET, то GET-запит на resource визве цей метод. Якщо підресурс вказує свій URI{id}, то повний шаблон шляху к підресурсу буде resource/{id} і він буде оброблювати запити типу resource/42;
* клас повинен мати конструктор об’явлений як паблік, для виклику його в рантаймі. В рантаймі повинно бути можливим надати всі вхідні параметри конструктора. В якості параметрів конструктора, можуть використовуватись параметри, помічені JAX-RS анотаціями, такими як @HeaderParam, @PathParam, @FormParam.

**jQuery -** популярна JavaScript-бібліотека з відкритим сирцевим кодом. Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків[13]. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

jQuery можна підключити двома способами. Перший спосіб – завантажити потрібну бібліотеку з офіційного сайту та підключити її наступним чином: <script type="text/javascript" src="jquery.js"></script>. Другий спосіб, завантажувати бібліотеку при загрузці сторінки. Коли користувач переходить на сайт, бібліотека завантажується з сайту в браузер та зберігається там, тобто, при повторному переході на сайт, вона не буде знову завантажуватись, а буде братися з браузера. можна підключити таким чином: <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*37*

БДФЕУ.160067.004.ПЗ

</script>

**AJAX -** підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX — це не самостійна технологія, а швидше концепція використання декількох суміжних технологій[14].

UI в проекті створювався за допомогою **Bootstrap Framework**. Bootstrap - це безкоштовний набір інструментів з відкритим сирцевим кодом, призначений для створення веб-сайтів та веб-застосунків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-застосунків.

Bootstrap – це клієнтський фреймворк, тобто інтерфейс для користувача, на відмінно від серверної сторони, яка знаходиться на сервері. Репозиторій з даним фреймворком є одним з найбільш популярніших на GitHub. Серед інших, його використовують NASA та MSNBC.

**Chartjs** – JavaScript-бібліотекою, за допомогою якої можна створювати різні типи діаграм, при цьому використовуються можливості HTML5-елементу – canvas[16].

**Underscore.** В його склад входять більше 80 функцій, які спростовують роботу з колекціями, функціональними зв’язками, javascript-шаблонізаціями та іншими.

* 1. **Реалізація та UML-діаграма.**

Реалізація проекту була проведена в декілька етапів:

* ініціалізація:
  + - аналіз ідеї, розробка проектного завдання;
    - визначення цілей, обмежень і пріоритетів проекту;
* планування:
  + - визначення об’єму проекту;
    - виявлення всіх можливих ризиків, пов’язаних з реалізацією проекту і розробка способів їх вирішення;
* реалізація:
  + - створення кінцевого варіанту продукту;
    - ВИРІШЕННЯ проблем що виникли.

Користувацький інтерфейс(UI) – оснований на Bootstrap Framework. Весь інтерфейс був побудований на бутстрапівській системі сітки. Головної її перевагою є те, що за її допомоги досягається повна адаптивність сайту. Система сіток основана на 12-х колонках, які масштабуються відповідно до розміру пристрою або до розміру оглядового вікна. В проекті знаходиться єдиний html файл, які містить в себе всі контейнери: Home, Settings, Charts, About та інші.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*38*

БДФЕУ.160067.004.ПЗ

Всі переходи, анімації, були створенні за допомогою jQuery. Запити до REST API що знаходиться на сервері веб проекту здійснюються за допомогою AJAX, задля уникання лишніх завантажень/перезавантажень сторінки.

Створене REST API приймає такі методи як GET, PUT, POST, DELETE. Коли запит надходить до ресту, він аналізує його та передає керування на відповідно анотований метод. Цей метод стягує з запиту за допомогою анотацій @FormParam потрібні параметри які були реалізовані та відправленні. Далі ці дані опрацьовуються, та передаються в сервісний рест клас в якому, в залежності від анотованого методу, викликається відповідний метод, а якому, якщо надходить лиш одна дата, передає цей параметр в інший клас, в якому знаходиться метод getNeededDateList() котрий шукає дані на потрібне число в базі даних. Якщо він їх знаходить – повертає список типу Rates, який містить потрібні курси, далі вони повертаються на метод з класу Resources, і, відповідно цей метод вертає дані в форматі JSON на клієнтську частину, де вони обробляються за допомогою JavaScript та виводяться в потрібному місці. Якщо ж курсів на потрібну дату не існує в базі даних – управління передається в клас Parser, там відкривається конекш і йде call до API потрібного банку. Після цього перевіряється повернений статус код запиту. Якщо він рівний 200 – вертається об’єкту класу HttpURLConnection, який передається в інший метод для парсингу відповіді. Там зразу перевіряється відповідь. Якщо отриманий масив пустий – викликається клас WebApplicationException, який зразу виходить з методу і вертає на ресурс метод дані помилки в вигляді JSON, вони передаються на кліент, і в функції error виводяться за допомогою бібліотеки Notify.js як пуш повідомлення на екран. Якщо ж масив не пустий, дані парсяться, записуються БД та відправляються в ресурс. Далі по аналогії.

Всі таблички з проекту виводяться за допомоги шаблонізатора underscore.js.

Всі графіки будуються за допомогою бібліотеки Chartjs, а дані для побудови оброблюються практично по аналогії до даних для таблиць, лиш перевіряються додаткові умови, наприклад, якщо потрібні курси НБУ то перевіряються дні тижня, якщо це не субота або неділя, або дні, на які офіційно припадають вихідні. Якщо в діапазон дат входять такі дні – вони ігноруються, тобто по ним не відбувається запит до API банку.

*Зм.*

*Аркуш*

*№ докум.*

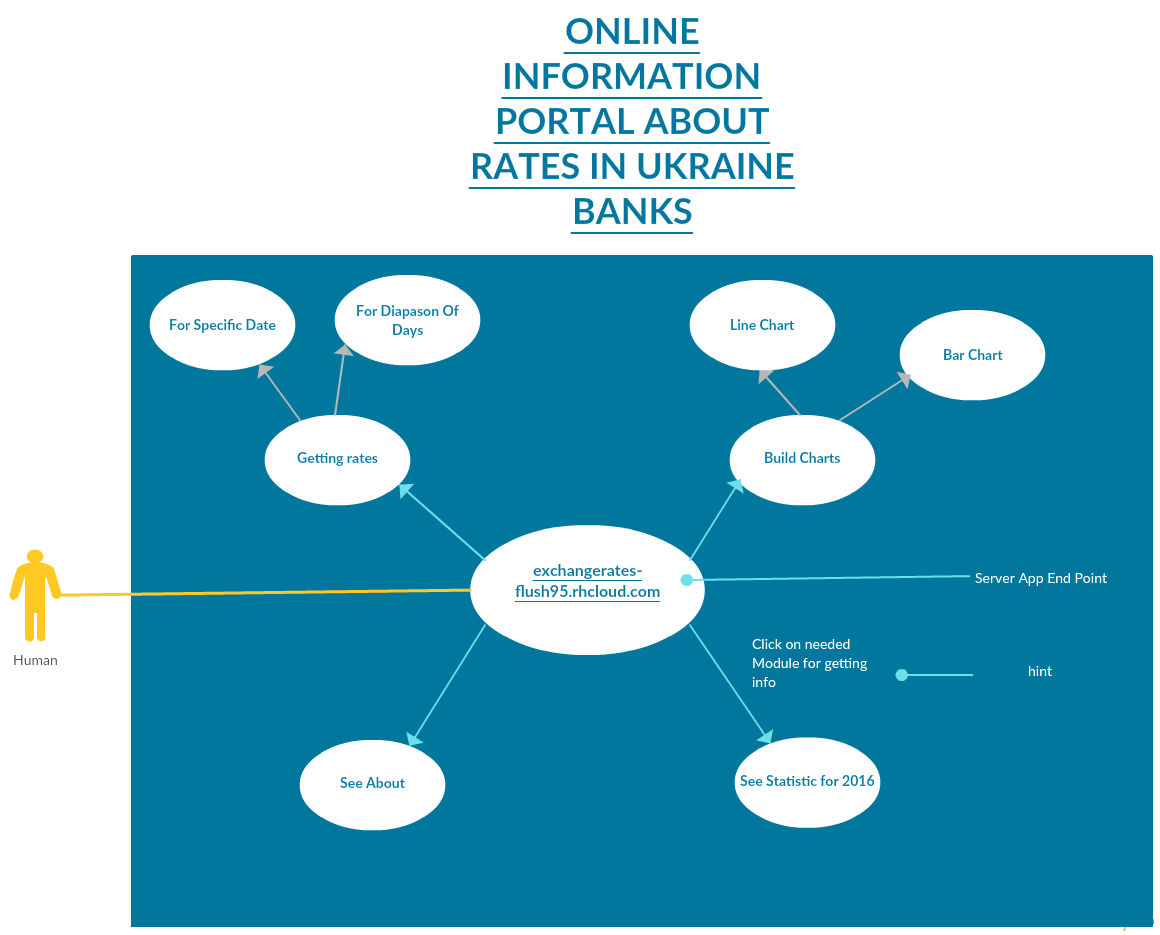
*Підпис*

*Дата*

*Арк.*

*39*

БДФЕУ.160067.004.ПЗ



*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

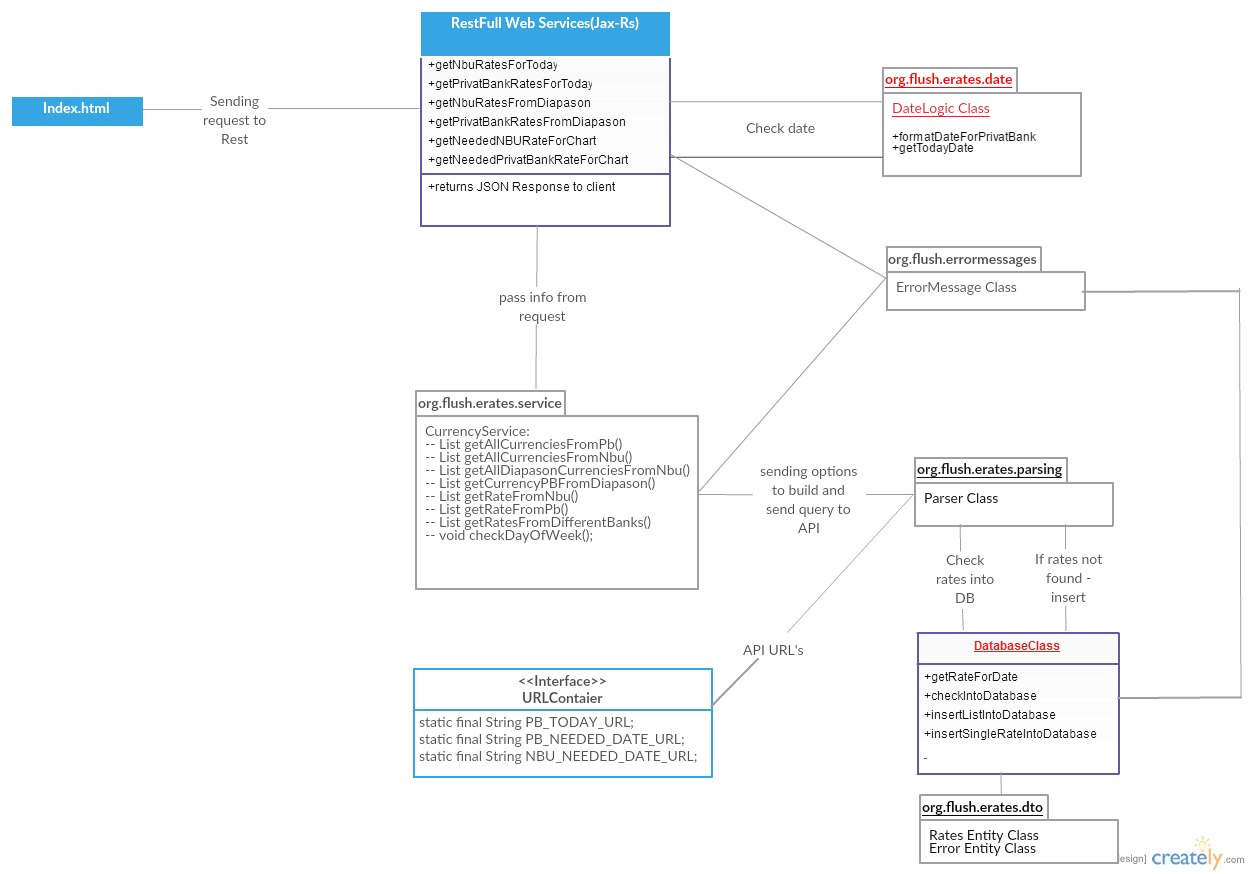
*Дата*

*Арк.*

*40*

БДФЕУ.160067.004.ПЗ

Рисунок 3.2.1 – Візуальне представлення модулів (Activity Diagram)



*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*41*

БДФЕУ.160067.004.ПЗ

Рисунок. 3.2.1 Діаграма класів.

РОЗДІЛ 4

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*42*

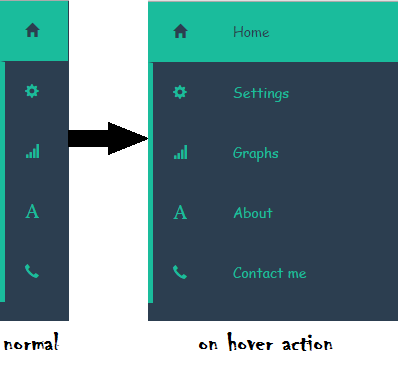
БДФЕУ.160067.004.ПЗ

ЗАСТОСУВАННЯ РОЗРОБЛЕННОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПРАКТИЧНИХ ЦІЛЕЙ

**4.1. Застосування інформаційної системи для курсів валют.**

При розробці інформаційної системи для аналізу валютних коливань були використані принципи компонентної розробки. Основні модулі проекту:

* Menu – відображається в лівій стороні екрану. При наведенні відкривається (рис. 4.1).
* Home – головна сторінка проекту. На ній відображаються:
  + - форми для побудови таблиць по сьогоднішній даті та по діапазону дат (рис. 4.2);
    - таблиці які відображають курси (рис. 4.3).
    - компонент експорту таблиці в xls формат(рис 4.4).
* Chart – сторінка, на якій будуються графіки
  + - на ній присутня форма для задання параметрів побудови (рис. 4.5);
    - графік, який оснований на html5 canvas:
      * Line Chart (рис. 4.6);
      * Bar Chart (рис. 4.7).
    - компонент alert, який показує ріст курсу за потрібний період (рис. 4.8).
* Плаваючий компонент notify для відображення помилок введення даних (рис. 4.9).
* About – містить інформацію про проект (використані технології, ціль і тощо)(4.10).



*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

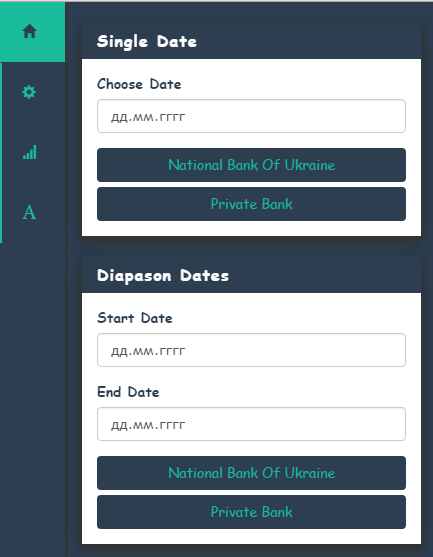
*Дата*

*Арк.*

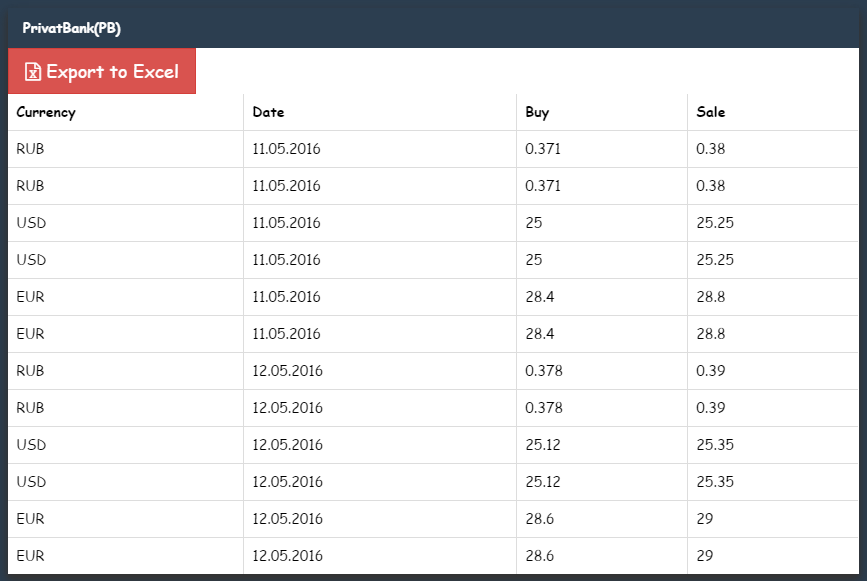
*43*

БДФЕУ.160067.004.ПЗ

*Рисунок 4.1. Модуль Menu додатку в нормальному стані та при наведенні.*



*Рисунок 4.2. Форми для побудови таблиць.*



*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

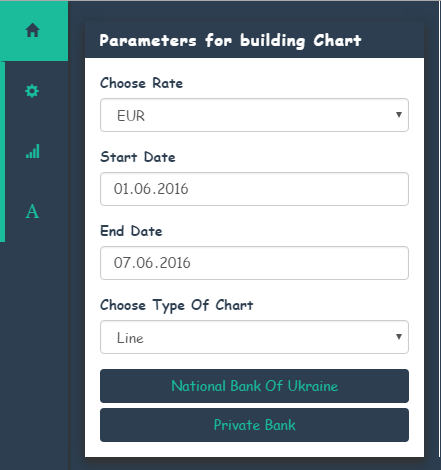
*44*

БДФЕУ.160067.004.ПЗ

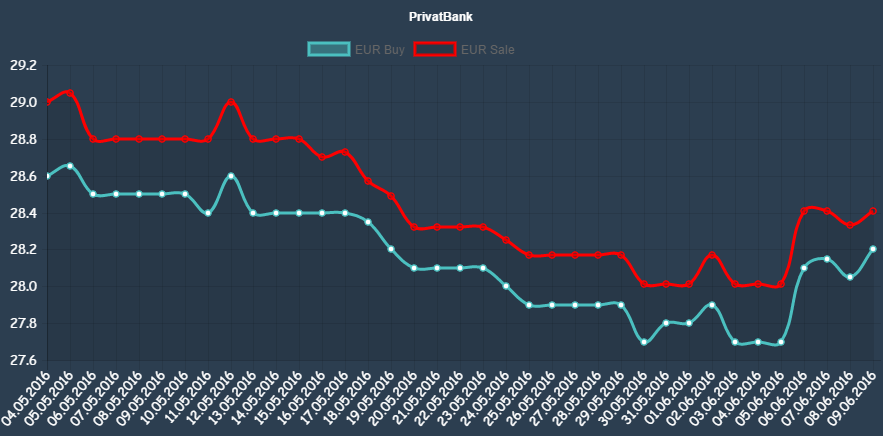
*Рисунок 4.3. Приклад побудованої таблиці.*

*s*

*Рисунок 4.4. Компонент експорту даних в xls формат*



*Рисунок 4.5. Форма для побудови графіків.*



*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

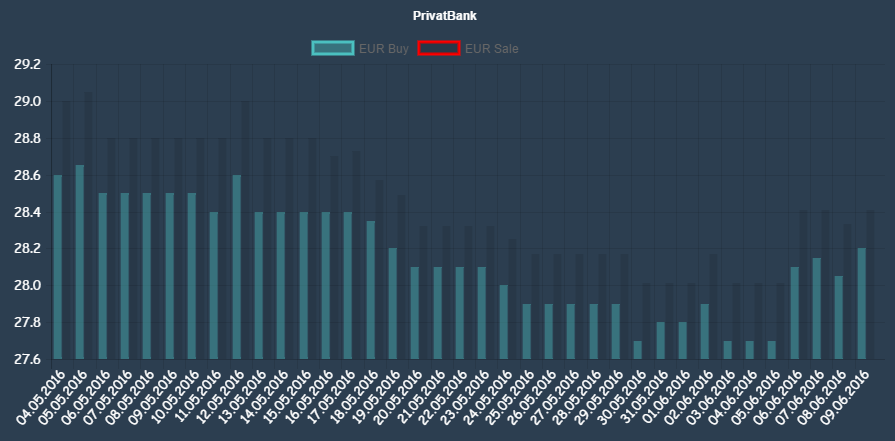
*Дата*

*Арк.*

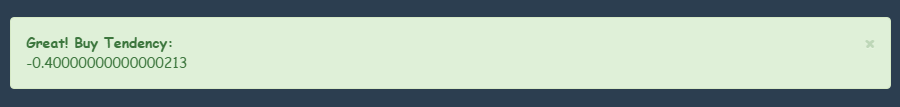
*45*

БДФЕУ.160067.004.ПЗ

*Рисунок 4.6. Line Chart(графік)*



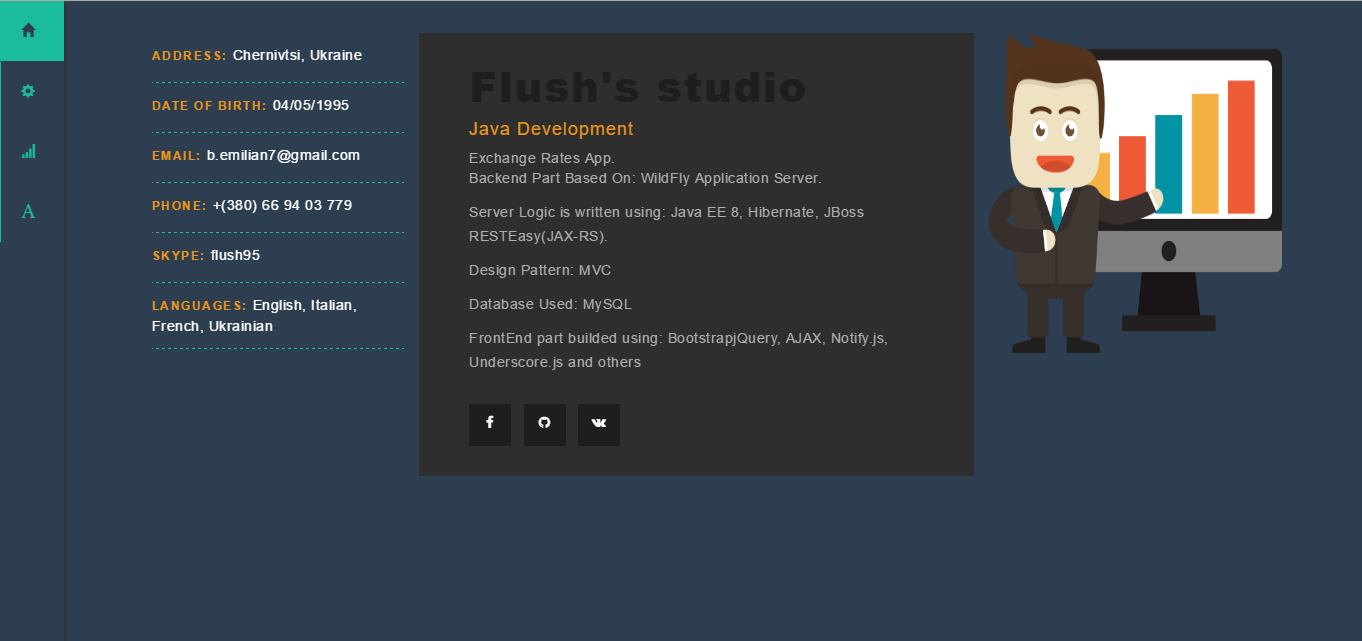
*Рисунок 4.7. Bar Chart*

******

*Рисунок 4.8. Компонент Alert*

*s*

*Рисунок 4.9. Компонент Notify*

**

*Рисунок 4.10. Компонент About*

Переходи між компонентами в меню здійснюються за допомогою jQuery, а саме за допомогою функцій *hide(), show(), slideUp(), slideDown().*

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*46*

БДФЕУ.160067.004.ПЗ

Для отримання курсів на поточне число достатньо скористатися компонентом «Single date». Для цього можна навіть не вказувати дату в полі.

Для отримання курсів на конкретне число потрібно вибрати в цьому же компоненті дату та натиснути на кнопку (з потрібним банком).

Щоб отримати курси на діапазон дат, потрібно заповнити компонент «Diapason Dates». Якщо залишити його пустим і відправити запит на отримання даних – в правій стороні вікна можна буде спостерігати notification блок(рис. 4.9) з відповідною помилкою.

Результати запитів курсів з модулю Home відображаються у таблицях (рис. 4.3).

Для побудови Line графіку (рис. 4.6) потрібно заповнити компонент «Parameters For Building Charts» (рис. 4.5), а саме заповнити поля з датами, вибрати потрібну валюту (долар, євро, рубль) та натиснути кнопку з потрібним банком. Результати відобразяться в компоненті html5 – canvas, який містить в себе iframe. На графіці присутні ціни на купівлю та продаж валюти. При бажанні можна натиснути наприклад на Eur Buy, для того щоб не відображати ціну купівлі. Графіки перебудовуються автоматично. Не потрібно перезавантажувати додаток. Таке ж діє і для Bar графіку.

Під графіком відображається компонент alert який відображає на скільки змінилася ціна з початку діапазону й до кінця. При бажанні можна його видалити, натиснувши на хрестик.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*47*

БДФЕУ.160067.004.ПЗ

Для експорту отриманої таблиці в xls формат потрібно натиснути на компонент Export to Excel (рис. 4.4). Конвертований файл завантажиться по замовчуванню за шляхом С://User/Downloads.

ВИСНОВКИ

Результатом виконання дипломного проекту є розробка програмного продукту, призначеного для проведення аналізу валютних коливань.

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*48*

БДФЕУ.160067.004.ПЗ

Розроблений програмний продукт є актуальним для людей, яким треба швидко та точно дізнатися курс валют в банках України на сьогоднішній день, на конкретну дату чи на діапазон дат, також для тих, хто проводить різні дослідження росту або падіння ціни на певну валюту, хоче отримати певні статистичні дані.

В результаті був створений веб-сайт (додаток), який дає змогу проаналізувати тенденцію валютних курсів. Проект розроблений за допомогою новітніх технологіях і фреймворків: WildFly application server, Java EE, Hibernate Framework, Maven, RESTfull Web сервіси. Весь фронтент побудований за допомогою Bootstrap Framework. Також використані: бібліотека для побудови графіків – ChartJs, шаблонізатор – Underscore.js для спрощення багатьох аспектів виводу даних, Notify.js для виводу кастомних помилок та технологія AJAX, яка дає змогу відправляти данні без перезавантаженя сторінки.

В результаті виконання дипломного проекту було:

1. розроблено веб-сайт «ExchangeRates», який надає можливість проводити статистичні дослідження даних;
2. в програмному продукті передбачено можливість отримання візуального представлення оброблених даних за допомогою побудови таблиць та графіків.
3. дана розробка володіє можливістю надання даних за введеними датами;
4. простота в користуванні, яка забезпечується мінімалістичним UI, що є досить зручно;
5. швидкодія отримання потрібних результатів є досить високою, що не можливо не відмітити, як позитивну сторону розробленого додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*49*

БДФЕУ.160067.004.ПЗ

1. Артус М. Гроші та кредит: Навчальний посібник/ Мирослав Артус,; Європейський ун-т. - К.: Вид-во Європейського ун-ту, 2004. - 161 с.
2. Гроші та кредит: Підручник/ За заг. ред. М.І. Савлука,; М-во освіти України. Київський нац. економ. ун-т. - К.: КНЕУ, 2001. - 602 с.
3. Гроші та кредит: Підручник для вузів за спец. «Фінанси та кредит»/ М. Савлук, К. Зуллас, А. Коряк. - К.: Либідь, 2002. – 330 с.
4. Гроші та кредит: Підручник/ За ред. Богдана Івасіва,; М-во освіти і науки України, НБУ, Терноп. акад. нар. госп.. - Тернопіль: Карт-бланш, 2000. - 510 с.
5. Oracle. Java EE. [Електронний ресурс] – Режим доступу: http://www.oracle.com/technetwork/java/javaee/overview/index.html. Дата звернення: 28.05.2016.
6. Hibernate ORM. [Електронний ресурс] – Режим доступу: http://hibernate.org/orm/. Дата звернення: 28.05.2016.
7. JSON. [Електронний ресурс] – Режим доступу: http://www.json.org/. Дата звернення: 29.05.2016.
8. Шевчук А. jQuery для начинающих. [Електронний ресурс] – Режим доступу: http://anton.shevchuk.name/jquery-book/. Дата звернення: 29.05.2016.
9. Chris Fehily. SQL Database Programming./ Chris Fehily. – М.: ДМК Прес, 2012. – 323 c.
10. Madhusudhan Konda. Just Hibernate (Covers 4.0)./ Madhusudhan Konda – М.: ДМК Прес, 2014. – 289 c.
11. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual (англ.). [Текст] – 284 с.
12. JavaScript diagramming library [Електронний ресурс] – Режим доступу:

http://www.jointjs.com

1. Predictable state container for JavaScript apps [Електронний ресурс] Режим доступу: http://redux.js.org.
2. A JavaScript library for building user interfaces [Електронний ресурс] – Режим доступу: https://facebook.github.io/react

*Зм.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*50*

БДФЕУ.160067.004.ПЗ

1. Predictable state container for JavaScript apps [Електронний ресурс] – Режим доступу: http://redux.js.org.
2. A JavaScript library for building user interfaces [Електронний ресурс] – Режим доступу: https://facebook.github.io/react.