**Міністерство освіти і науки України**

**Харківський національний університет ім. В. Н. Каразіна**

Факультет комп’ютерних наук

Кафедра моделювання систем і технологій

ЗАТВЕРДЖЕНО

Завідувач кафедри МСіТ

\_\_\_\_\_\_\_\_ Микола ТКАЧУК

**Звіт**

про виробничу практику

|  |  |
| --- | --- |
| Оцінка \_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Члени комісії:  \_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) (прізвище, ініціали)  \_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) (прізвище, ініціали)  \_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) (прізвище, ініціали)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р. | Виконав: студент 3 курсу, групи КС- 32  спеціальності: 122 «Комп’ютерні науки»  \_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) (прізвище, ініціали)  Керівник від бази практики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ступень, звання, прізвище та ініціали)  Рекомендована оцінка: \_\_\_\_\_\_\_\_ балів  \_\_\_\_\_\_\_\_\_\_\_\_   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) (прізвище, ініціали) |

Харків – 2021

**Зміст**

[Завдання на виробничу практику 3](#_Toc77943144)

[Вступ 5](#_Toc77943146)

[Програма виробничої практики. Індивідуальне завдання 6](#_Toc77943147)

[Основна частина 7](#_Toc77943148)

[Висновки 16](#_Toc77943149)

[Використані джерела 17](#_Toc77943150)

**Міністерство освіти і науки України**

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

Кафедра моделювання систем і технологій

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**на виробничу практику студентів третього курсу**

Студента  Обори Романа Сергійовича

(прізвище, ім’я, по батькові студента)

Група  КС-32   E-mail:   oboraroman2018@gmail.com

Т.м.:   +380979690125

(студента)

Назва підприємства   Ідеясофт Солюшнс

(назва організації, фірми, ТОВ, ПП, інше)

Керівник практики від підприємства   Вишнякова Єлизавета Григорівна

(прізвище, ім’я, по батькові повністю)

Т.м.:   +380933552886

(керівника практики від підприємства)

Термін виробничої практики: з 28.06.2021 р. до 18.07.2021 р.

1. **Завдання на термін проходження практики**

  Підготувати технічну проектну документацію про запуск, налаштування

  проекту веб-додатку для роботи з синхронізацією транзакцій на

  виділених серверах, опис ключових використовуваних фреймворків та

  подробиці роботи, можливостей і використання додатку.

1. **Календарний графік проходження практики**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Пор.**  **№** | **Назви робіт, що має виконати студент** | **Тижні проходження практики** | | | | |
| 1 | 2 | 3 | 4 | 5 |
|  | Знайомство з підприємством. | ✓ |  |  |  |  |
| 2 | Проходження інструктажів з техніки безпеки та охорони праці. | ✓ |  |  |  |  |
| 3 | Отримання завдання. | ✓ |  |  |  |  |
| 4 | Знайомство з літературою то інтернет-джерелами. | ✓ | ✓ |  |  |  |
| 5 | Знайомство з проектною документацією | ✓ | ✓ |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Пор.**  **№** | **Назви робіт, що має виконати студент** | **Тижні проходження практики** | | | | |
| 1 | 2 | 3 | 4 | 5 |
| 6 | Налаштування проекту для роботи на локальному комп’ютері. |  | ✓ |  |  |  |
| 7 | Знайомство з можливостями та flow додатку. |  | ✓ |  |  |  |
| 8 | Розробка додатку |  |  | ✓ |  |  |
| 9 | Підготування технічної документації з запуску проекту та опису використовуваних фреймворків. |  |  | ✓ |  |  |
| 10 | Підготовка звіту з виробничої практики |  |  | ✓ |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Підпис студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис керівника практики Підпис керівника практики

від кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_ від бази практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Вступ

У якості місця проходження практики було обрано компанію по розробці програмного забезпечення IdeaSoft Solutions, яка нещодавно стала членом корпорації Sigma Software. Профілем діяльності компанії є розробка програмних рішень у напрямках Backend/Frontend розробки, розробки мобільних додатків, штучного інтелекту і машинного навчання, рішень в області блокчейну, а також UI/UX дизайну з використанням різноманітних технологій, зокрема Angular, Node JS, Java, Kotlin, .NET, C++, Blockchain, MySQL та ін.. IdeaSoft Solutions на даному етапі свого розвитку має понад 250 успішно супроводжуваних проектів, і понад 120 досвідчених фахівців.

# Програма виробничої практики. Індивідуальне завдання

Завданням на термін проходження виробничої практики є розробка розподіленого веб-додатку для роботи з транзакціями. Система має бути розподілена на backend та frontend частини і забезпечувати синхронну обробку вхідних (incoming) та вихідних (outgoing) транзакцій, контроль балансу компаній, а також захист інформації шляхом авторизації користувачів в системі. Кінцева точка REST для надання інформації про поточний баланс має бути прихована і використовуватись тільки для внутрішніх API-викликів між backend та frontend рівнями.

В процесі розробки мають бути використані такі технології:

* Віддалені сервери Amazon AWS instances з Ubuntu 20.04. Backend та Frontend частини повинні бути запущені на різних серверах, між якими має бути налагоджений зв’язок.
* Backend-частина має бути написана мові Java з використанням фреймворку Spring Boot.
* Для зв’язку додатку з frontend має використовуватись Spring Web.
* Під’єднання до бази даних на MySQL-сервері повинно відбуватися за допомогою Spring Data.
* Налаштування доступу, безпеки, авторизації має бути підключено і налаштовано через Spring Security.
* Frontend-частина має бути написана на мові TypeScript з використанням фреймворку Angular.
* Повинна бути можливість перегляду веб-інтерфейсу backend (наявних для запуску транзакцій кінцевих точок REST) за допомогою фреймворку Swagger.

# Основна частина

Розробка проекту була розпочата з огляду інтернет-джерел та пошуку інформації про необхідні для розробки технології. Деякі моменти довелося вивчати каскадно, зокрема права користування та можливості віддалених Amazon AWS серверів, їх налаштування та корегування, встановлення та зняття обмежень вхідного/вихідного трафіку на порти.

Процес розробки додатку розпочався з створення виділених серверів. Було вирішено розбити додаток на 3 сервери: один буде працювати у якості MySQL-серверу і зберігати дані додатку, інший – виконувати логіку програми і управляти даними (backend), а останній – виконувати роль рівня представлення, відображаючи необхідні дані та надаючи інтерфейс для роботи з ними (frontend).

На AWS instance, який відповідав за рівень моделі було встановлено MySQL сервер і створено базу даних firm\_balances. База даних має зберігати інформацію про поточний баланс кожної з компаній і транзакції, що надходять (сума та фірма, якій адресована). Крім рівня зберігання даних, було вирішено також додати перевірку на рівні бази даних вхідних даних (для більш високого рівня захисту додатку). Було створено тригер, який виконується перед записом наступної транзакції в базу і перевіряє суму транзакції: вона не повинна перевищувати поточний баланс фірми. Якщо сума задовільна, то тригер відповідним чином змінює баланс вказаної фірми та виконує запис транзакції.

Також, спеціально для додатку на сервері був створений новий профіль користувача бази даних, за допомогою якого буде під’єднуватися до БД backend-рівень.

Після того, як MySQL-сервер було налаштовано на доступ через мережу, розпочався процес написання backend-частини додатку. Каркас проекту у збірці Maven з необхідними залежностями (Spring Boot,Web, Data, Security) був згенерований за допомогою ресурсу Spring Initializer.

##### В першу чергу були створені класи-сутності (Entity) до відповідних таблиць в базі даних. З їх допомогою буде можливо не тільки виконання запису/читання до БД, а й обмін інформацією з frontend-рівнем за допомогою JSON-серіалізації.

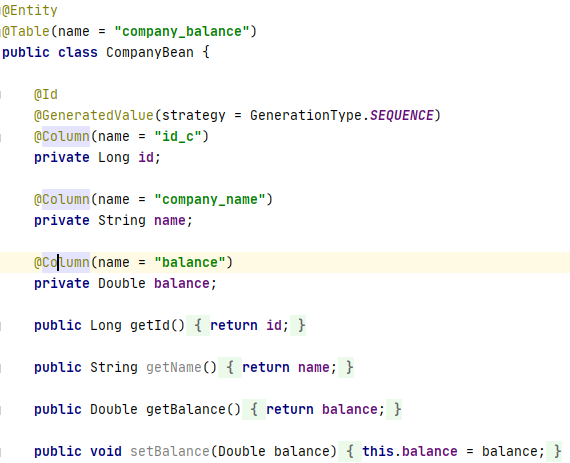
**

Рисунок 1 - приклад Entity-класу

##### Далі почалася розробка класів-сервісів, які буть слугувати REST-контролерами. Ці класи описують веб-інтерфейс додатку, на який йому можливо посилати HTTP-запити від frontend-частини або swagger UI. У додатку є два основних сервіси: отримання GET-запитом даних про поточний баланс компаній для відображення (/balance), та надсилання POST-запитом даних нової транзакції (/transaction). Для виконання операцій вставки/читання з БД в відповідних сервісах є поля-репозиторії, які ми автогенеруюємо за допомогою Spring. За допомогою анотації @RequestMapping вказуємо який шлях запиту обробляє контролер. Спочатку вказуємо версію програми, потім – назву кінцевої точки. Методи, які будуть обробляти конкретні HTTP-запити вказуємо відповідні анотації. За необхідності в них можна вказати додатковий шлях: нам це знадобилося для розділення ендпоінтів incoming і outgoing транзакцій (усі запити транзакцій починаються з “/v1/transaction/”, а кінцева точка обирається відповідно до того, що йде після цього: “/v1/transaction/ incoming” чи “/v1/transaction/outgoing”).

##### За допомогою параметрів consumes/produces вказуємо який тип даних ми отримуємо на вході та на виході, тобто які дані ми отримуємо/передаємо через HTTP-запит. В нашому випадку це JSON. Spring самостійно буде сереалізувати /десереалізувати java-об’єкти в JSON.

##### Для того, щоб запити, які приходитимуть на ці кінцеві точки не блокувались CORS Policy, необхідно анототувати контролери на @CrossOrigin з адресою та портом з яких будуть приходити запити (тобто адресою frontend-серверу). Оскільки в постановці задачі вказано приховати ендпоінт балансу,позначаємо його анотацією @ApiIgnore. Це приховає його від неавторизованих користувачів і заодно від swagger.



Рисунок 2 - REST-контролер балансу

##### Контролер транзакцій влаштований складніше. По перше, він обслуговує два ендпоінти: на incoming і outgoing транзакції. По друге, він отримує дані через POST-запит. Для того, щоб вони автоматично трансформувались в сутності з JSON вказуємо біля вхідного параметру анотацію @RequestBody, а для того, щоб відправити їх назад до рівня представлення також у вигляді JSON – анотуємо метод @ResponseBody.

##### У випадку incoming транзакції ми записуємо дані в БД, а якщо виконується outgoing транзакція – виконуємо перевірку. У разі якщо дані не пройшли перевірку, ми маємо якось інформувати про це рівень представлення. Для цього вирішено було створити спеціальне виключення. Позначивши його анотацією @ResponseStatus(HttpStatus.UNPROCESSABLE \_ENTITY), ми вказуємо, що в разі виникнення цього виключення Spring Web у якості відповіді на запит має відправити HTTP-статус Unprocessable Entity, тобто невірні вхідні дані.

##### Далі ми переходимо до основного призначення даного завдання – синхронізації транзакцій. За умовою завдання потрібно зробити так, щоб декілька одночасних транзакцій не змогли зіпсувати дані одночасним записом у БД. В Spring Data для цього існують спеціальні засоби у вигляді анотації @Transactional. Метод який позначено даною анотацією може виконуватись тільки синхронно, тобто якщо інший потік спробує виконати запит в той час коли один вже почав роботу з БД, він отримає код помилки HTTP Internal Server Error. Транзакцію до БД можна налаштувати за допомогою параметрів анотації. По-перше, був встановлений режим розповсюдження транзакцій REQUIRES\_NEW (кожен запит створює нову транзакцію якщо в даним момент якась вже існує, не виконує дії в інсуючій), а по-друге, встановлено рівень ізольованості SERIALIZABLE, який розташовує всі транзакції по порядку і робить чітку послідовність не перекриваючих один одного викликів. За допомогою цих засобів вдалося синхронізувати запити до БД.

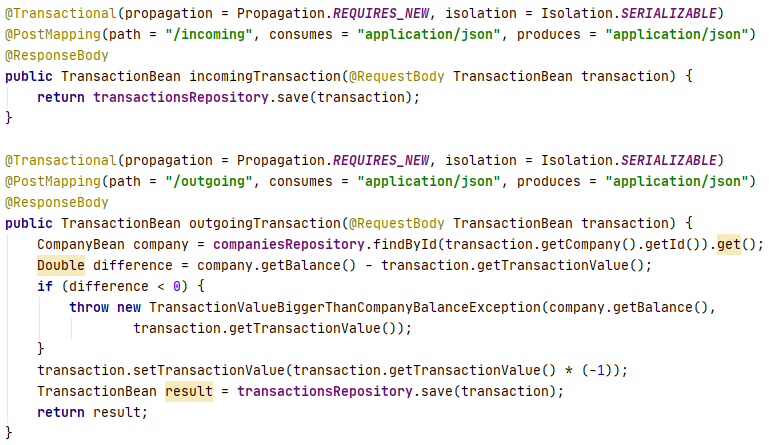


Рисунок 3 - обробка транзакцій

##### Далі було проведено конфігурацію авторизації і створення облікового запису для користувача програми. За допомогою засобів Spring Security:



Рисунок 4 - конфігурація Spring Security

##### Далі переходимо до створення frontend проекту. Його скелетон генеруємо автоматично за допомогою утиліти ng. По-перше, потрібно налаштувати frontend на працю в режимі proxy по відношенню до backend за допомогою конфігурацій проекту в angular.json. Далі створюємо класи-сервіси, які будуть відправляти запити до рівня управління та реєструємо їх. Оскільки об’єкти компаній які ми отримаємо від рівня управління нам буде зручно використовувати і на рівні представлення, створимо також додатковий відповідний інтерфейс і параметризуємо метод get масивом цих об’єктів.



Рисунок 5 - frontend-сервіс отримання компаній

##### При виборі типу транзакції до шляху буде додаватися відповідна кінцева точка REST.

##### 

Рисунок 6 - frontend-сервіс відправки транзакцій

##### Після цього можна вносити логіку роботи до класу AppComponent – основного компоненту програми який буде запущений одночасно з html-сторінкою і через який користувач буде викликати основний функціонал. При завантаженні цього класу буде генеруватися підписка на повернення результату HTTP-запиту на ендпоінт балансу, у якій отримані об’єкти компаній вноситимуться у поле компоненту.

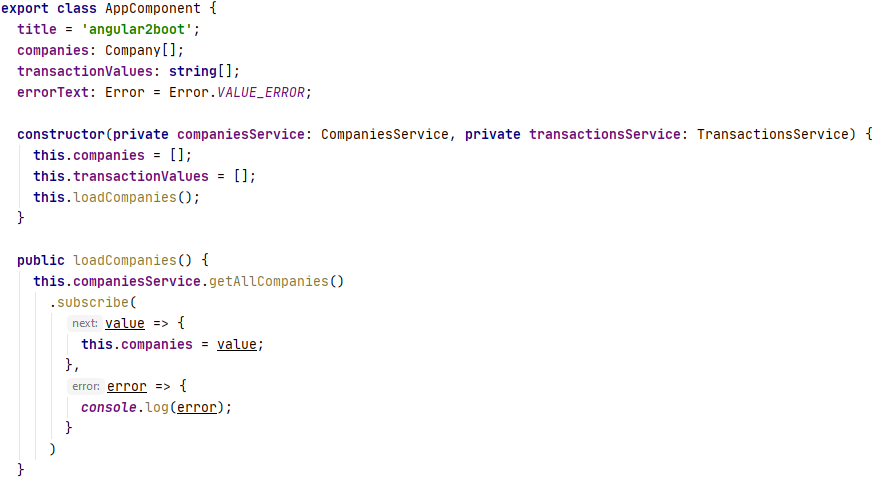


Рисунок 7 - дії при завантаженні сторінки компонентом

##### Додатково в цьому класі створюємо інтерфейс для відправки транзакцій – поліморфний метод post, у якому виконуємо перевірку введених даних та, використовуючи сервіс транзакцій, формуємо POST-запит до рівня backend. У підписку на код відповіді HTTP додаємо обробники зміни балансу у разі успіху, а також кодів помилок – у разі Unprocessable Entity інформуємо користувача про невірну суму транзакції, а у разі Internal Server Error – намагаємося повторити запит деяку кількість разів.

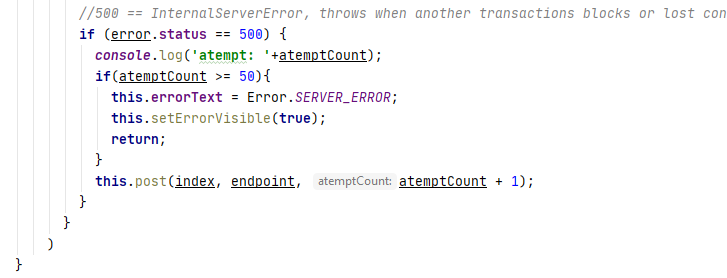
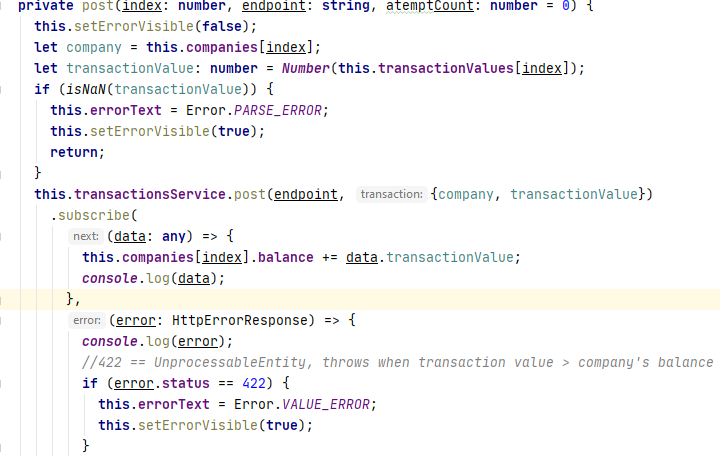


Рисунок 8 - метод генерування транзакції

##### Після цього була сформована веб-сторінка інтерфейсу з підключення необхідних викликів з використанням елементів і директив фреймворку Angular.



Рисунок 9 - розмітка основної веб-сторінки додатку

##### Розробку додатку на цьому було завершено.

# Висновки

##### В ході виконання практики було створено програмне забезпечення з декількома рівнями абстракції, розподілене на декільком віддалених серверах. Додаток має змогу виконувати синхронізовані запити і генерувати транзакції, які не забруднюють дані.

##### В процесі підготовки до завдання було отримано багато корисних знань в області конфігурування та адміністрування віддалених серверів AWS, налаштувань MySQL-серверу в мережі, налаштувань зв’язку різних рівнів додатку, розробки, збірки та запуску Spring Boot та Angular додатків, корегування підключень фреймворків Spring Data JPA, Security, Web, підключення інших проектів як модулів до основного з метою їх одночасної компіляції збірником Maven, налаштування безпеки та авторизації, а також ролей і доступу до конкретних ендпоінтів REST, приховання внутрішніх API-викликів, корегування налаштувань доступу до портів на серверах за допомогою AWS Security та брандмауера ufw та остаточного розгортання додатку на декількох серверах для успішної роботи.

##### В ході роботи був створений необхідний для виконання завдання програмний комплекс та покроковий опис кожного етапу розробки.

# Використані джерела

##### Tutorial Spring Security and Angular // URL: https://spring.io/guides/ tutorials/spring-security-and-angular-js/

##### Основы Angular: HTTP Client – Хабр // URL: https://habr.com/ ru/post/336280/

##### Angular2 and Spring Boot: Getting Started // URL: https://blog. jdriven.com/2016/12/angular2-spring-boot-getting-started/

##### Удаленные подключения к серверу базы данных MySQL // URL: https://andreyex.ru/bazy-dannyx/baza-dannyx-mysql/kak-razreshit-udalennye-podklyucheniya-k-serveru-bazy-dannyh-mysql/

##### Начало работы с Amazon AWS EC2 // URL: https://aws.amazon. com/ru/ec2/getting-started/

Керівник практики

від кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали) (підпис)