МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії

КУРСОВА РОБОТА

з дисципліни
“Основи програмування”
на тему:

"Розробка кроссплатформенного клієнт-серверного додатку для роботи груп користувачів над спільним проектом, а також комунікації між ними"

Студента 1 курсу, групи ІП-15
Мєшкова Андрія Ігоровича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник: ст.вик. Головченко М. М.

Кількість балів: _____
Національна оцінка _____

Члени комісії

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2022 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-15 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Мєшков Андрій Ігорович

(прізвище, ім'я, по батькові)

1. Тема роботи Розробка кроссплатформенного клієнт-серверного додатку для роботи груп користувачів над спільним проектом, а також комунікації між ними

2. Строк здачі студентом закінченої роботи 14 червня 2022

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 22 березня 2022

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	22.03.2022	
2.	Підготовка ТЗ	22.03.2022	
3.	Пошук та вивчення літератури з питань курсової роботи	05.04.2022	
4.	Розробка сценарію роботи програми	12.04.2022	
6.	Узгодження сценарію роботи програми з керівником	19.04.2022	
5.	Розробка (вибір) алгоритму рішення задачі	03.05.2022	
6.	Узгодження алгоритму з керівником	17.05.2022	
7.	Узгодження з керівником інтерфейсу користувача	17.05.2022	
8.	Розробка програмного забезпечення	31.05.2022	
9.	Налагодження розрахункової частини програми	07.06.2022	
10.	Розробка та налагодження інтерфейсної частини програми	08.06.2022	
11.	Узгодження з керівником набору тестів для контрольного прикладу	10.06.2022	
12.	Тестування програми	11.06.2022	
13.	Підготовка пояснювальної записки	12.06.2022	
14.	Здача курсової роботи на перевірку	14.06.2022	
15.	Захист курсової роботи	19.06.2022	

Студент _____

(підпис)

Керівник _____ Головченко М. М.

(підпис) (прізвище, ім'я, по батькові)

"___" _____2022 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 43 сторінки, 9 рисунків, 7 таблиць, 6 посилань.

Об’єкт дослідження: огляд існуючих програм для групової роботи користувачів над спільним проектом свідчить про існування певних недоліків. Основними недоліками всіх розглянутих додатків є відсутність мобільної версії, системи обміну повідомленнями, а також кроссплатформенності, що значно звужує сферу використання програм.

Мета роботи: створення кроссплатформенного клієнт-серверного додатку для усунення вище описаних недоліків. Програму може використовувати група людей, для зручної роботи над спільним проектом, студенти можуть використовувати додаток для виконання спільної лабораторної роботи.

Для вирішення поставленої задачі було обрано клієнт-серверну модель роботи. Для створення бази даних було обрано MySQL сервер, який разом із системою закодованої передачі даних забезпечують досить високу надійність, захищеність та гнучкість процесу обміну даними між користувачами та їх зберігання. Для створення клієнту було обрано мову програмування Java. Операційні системи для клієнта буди вибрані такі: Windows, Linux, MAC OS, Android, як найпоширеніші операційні системи для мобільних та настільних пристроїв.

ЗМІСТ

ВСТУП

Рівень розвитку сучасних технологій дозволяє обладнувати електронно-обчислювальні пристрої (персональні комп’ютери, кишенькові комп’ютери, смартфони) засобами доступу до глобальної мережі Інтернет. Це дає змогу не лише знаходити актуальну інформацію, але і обмінюватись нею в режимі реального часу — наприклад, спілкуватись із другом, який проживає в іншому куточку світу.

Інтернет (від англ. *Internet*) — всесвітня система взаємосполучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів. Інтернет також називають мережею мереж. Він складається з мільйонів локальних і глобальних, приватних, публічних, академічних, ділових і урядових мереж, пов'язаних між собою з використанням різноманітних дротових, оптичних та бездротових технологій. Інтернет становить фізичну основу для розміщення величезної кількості інформаційних ресурсів і послуг. [http://uk.wikipedia.org/wiki/Інтернет]

З появою мережі Інтернет з’явилися додатки, які дозволяють обмінюватися миттєвими повідомленнями в різних куточках світу. Програми, що використовують Інтернет, існують для будь-яких сучасних платформ. Вони досить гідно увійшли до нашого життя в усі сфери людської діяльності. Зараз складно уявити установу, яка не використовує спеціалізовані програмні додатки, які синхронізують інформацію за допомогою мережі Інтернет, нехай то буде університет, банк чи кондитерська фабрика. Інтернет використовують школярі, будівельники, військові, геологи і цей список можна продовжувати ще дуже довго.

Варто відмітити, що всі програми, які працюють через Інтернет можна розділити на два типи:

- Вузькоспеціалізовані – це програмні продукти розроблені для використання в певній сфері, наприклад, система управління банком, яка контролює роботу працівників та звітує про неї.

- Широкоспеціалізовані – допомагають користувачам використовувати Інтернет у власних цілях, наприклад, обмін повідомленнями, чи просто вивантаження або збереження інформації для доступу з любої точки світу.

Можна навести безліч прикладів, коли вкрай необхідна невелика проста система, але для групового використання, основним завданням якої був би контроль виконання певної роботи для групи осіб (з можливістю контролю стадії виконання проекту) і комунікація між ними. Наприклад, така система знадобилася групі розробників, які працюють над спільним проектом, або вчителям, які пишуть навчальний план. В режимі реального часу автори можуть писати чи редагувати власну книгу. Студенти можуть більш зручно працювати над командними лабораторними роботами. Викладачі можуть слідкувати за роботою студентів над певною роботою та вносити поправки в їх завдання і т.д. і т.п.

Таким чином, засоби що забезпечують взаємну роботу групи користувачів для роботи над спільним завданням, є дуже необхідними, а їх розробка і впровадження вкрай актуальним завданням.

Тому основна мета данної роботи – це створення кроссплатформенного клієнт-серверного додатку для роботи груп користувачів над спільним проектом і комунікації між ними, а також залучення нових виконавців для проекту.

Виходячи з мети роботи, були сформовані наступні завдання:

- Спроектувати систему на основі сучасних програмних засобів.
- Розробити клієнт-серверну модель системи.
- Реалізувати серверну частину системи, яка зберігає статичні та динамічні дані і забезпечує комунікацію між групами користувачів.
- Реалізувати клієнтську частину системи, яка надає інтерфейси для контролю і комунікації.
- Запустити і протестувати систему.

1 ПОСТАНОВКА ЗАДАЧІ

В процесі аналізу додатків було виявлено, що всі вони мають певні недоліки. Основними недоліками є відсутність комунікації, та мобільних версій. Також деякі версії аналогів не є кроссплатформенними, що не досить зручно при використанні таких систем як Linux чи MAC OS.

Таким чином, з перерахованого нижче випливає, що існують досить розвинені безкоштовні та платні програмні засоби для роботи над груповими проектами, але вони не вирішують сформульовану в даній роботі проблему. А саме:

- Відсутня система комунікації між групами користувачів.
- Відсутня кроссплатформенність.
- Відсутня мобільна версія для відстеження стану проекту, або для внесення поправок у завдання.

Отже, необхідно створити кроссплатформенне програмне забезпечення для роботи груп користувачів над спільним проектом, а також комунікації між ними. Для розширення комунікаційних можливостей програми інтегрувати сервіс обміну повідомленнями та групових дзвінків, а також сервіс для розшарування екрану.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Аналіз існуючих рішень.

2.1.1 2-plan project management software

2-plan project management software (рис. 2.1.1). Найпоширеніший додаток зі схожим функціоналом. Складається з трьох інструментів управління проектами. Перший – допомагає скласти розклад виконання завдань (безкоштовна версія). Другий – проектний менеджер. Останній – система управління завданнями.

Недоліки:

- Відсутність комунікації між користувачами.
- Відсутність мобільної версії.

Переваги:

- Синхронізація з GitHub.
- Побудова звітів про виконання роботи.

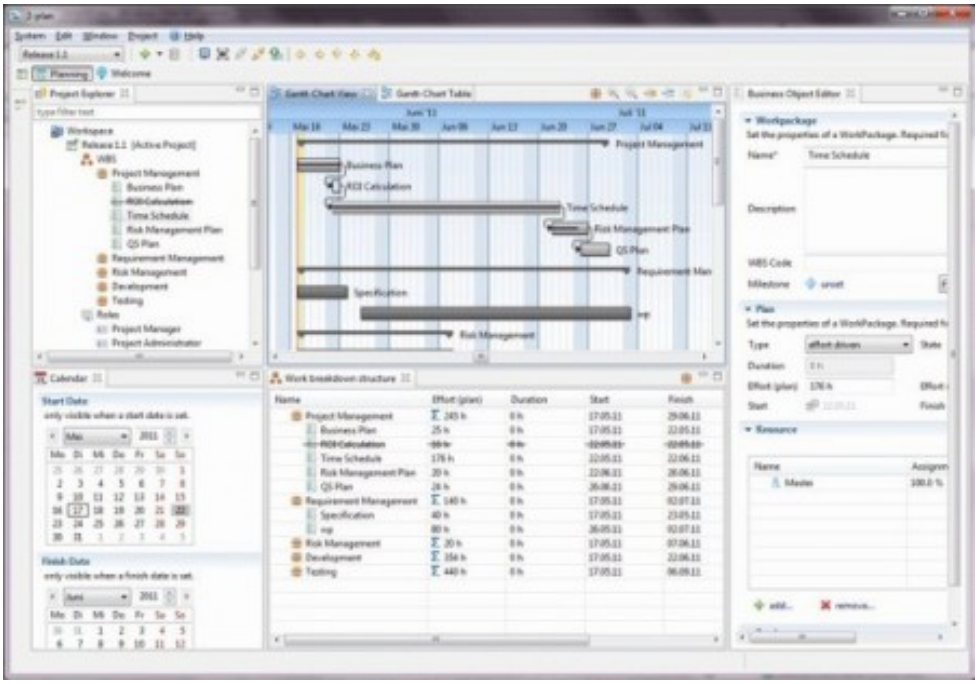

Рис. 2.1.1 – Вікно 2-plan project management software

2.1.2 BrightWork

BrightWork (Рис. 2.1.2) – надбудова для SharePoint. Створена для керування роботою проектів різного розміру і типу. Вона має поєднання шаблонів управління проектами та інфраструктури звітів SharePoint.

Недоліки:

- Відсутня мобільна версія.
- Відсутня система комунікації.

Переваги:

- Невелика за розміром.
- Не потребує потужних системних вимог.

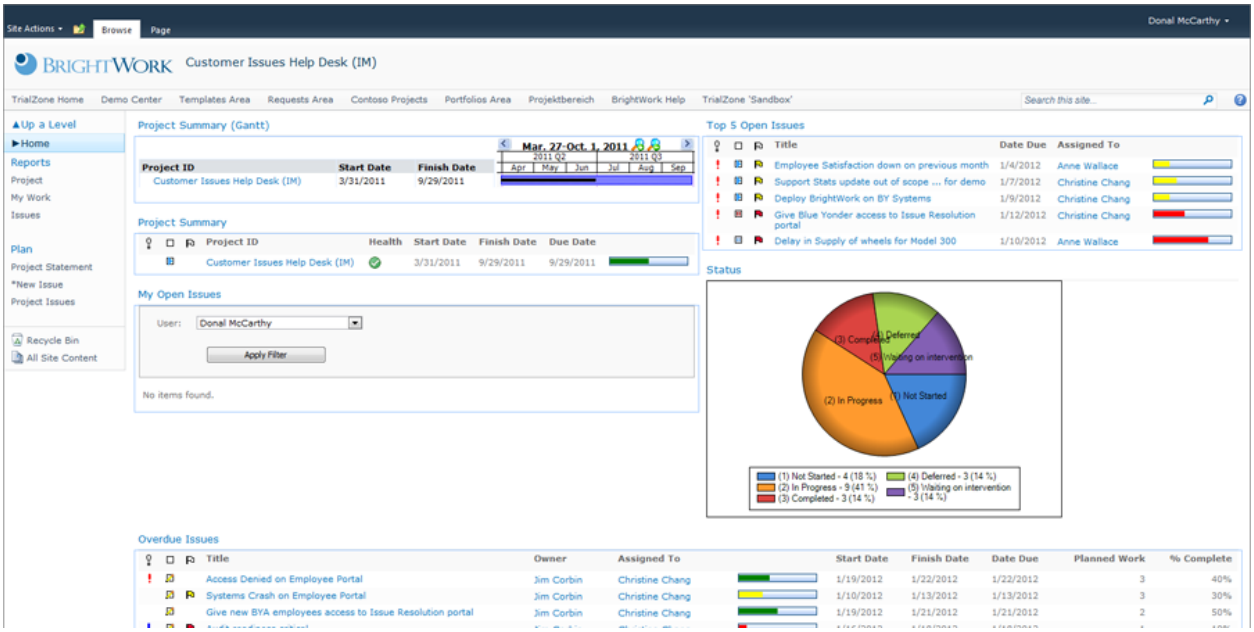

Рис. 2.1.2 – Вікно BrightWork

2.1.3 WorkPLAN

WorkPLAN (Рис. 2.1.3) – програмний продукт для керування не тільки окремими проектами, а й цілими підприємствами. Розроблений компанією Sescoi для використання виробниками або відділами, які працюють на основі проектів і потребують спеціалізоване програмне забезпечення для управління ними.

Недоліки:

- Велика вартість.
- Відсутність мобільної версії.

Переваги:

- Браузерний додаток.
- Система обміну миттєвими повідомленнями.

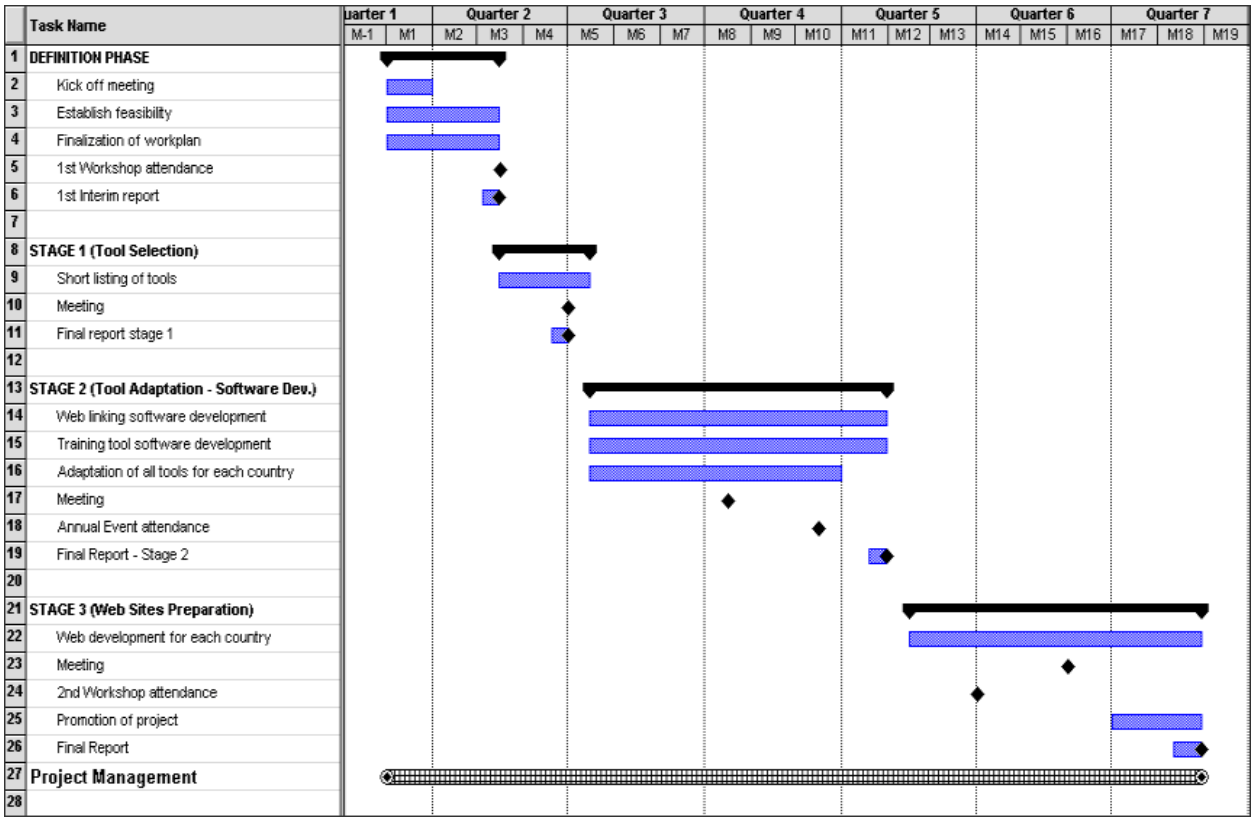

Рис. 2.1.3 – Вікно WorkPLAN

2.2 Програмні засоби

Для вирішення поставленої задачі була обрана клієнт-серверна модель системи наведена на рис. 2.2. Клієнт через Інтернет з'єднання підключається до сервера і виконує необхідні і допустимі для нього дії.

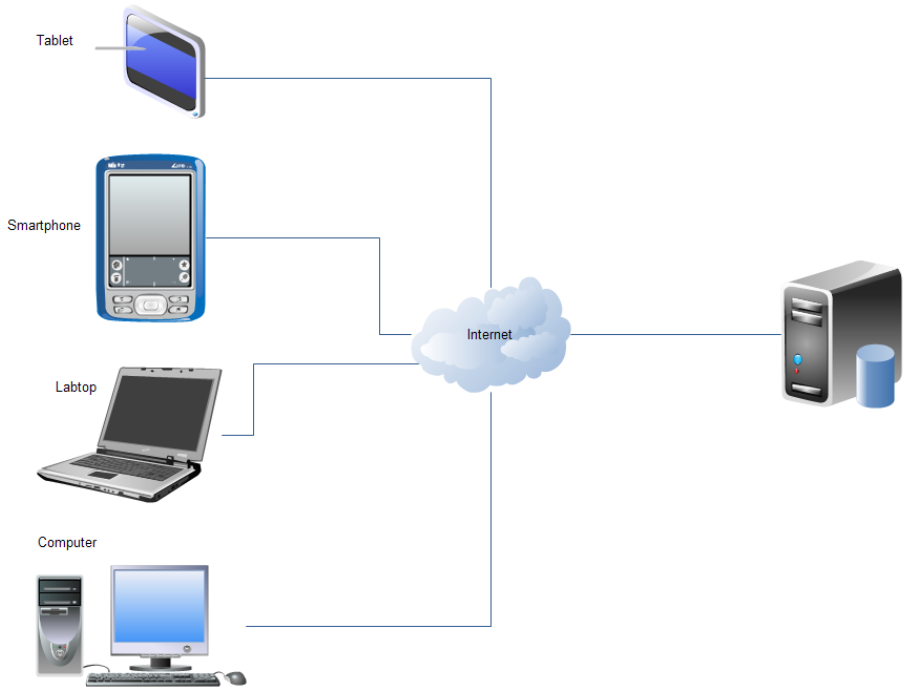

Рис. 2.2 – Принципова схема клієнт-серверної моделі

Для створення бази даних було обрано MySQL сервер, який разом із системою закодованої передачі даних забезпечує достатньо високу надійність, захищеність і гнучкість процесу обміну даними та їх зберігання. Для створення клієнта була використана мова програмування JAVA. Операційні системи були обрані такі як Windows, Linux, MAC OS, Android, iOS, Windows Phone, як найпоширеніші операційні системи. Для взаємодії сервера з клієнтом була вибрана технологія apache Mina.

MySQL – вільна система управління реляційними базами даних.

MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. [http://uk.wikipedia.org/wiki/MySQL]

Java (вимовляється Джава; інколи — Ява) — об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. [http://uk.wikipedia.org/wiki/Java]

Apache MINA – мережевий додаток, який дозволяє досить зручно і швидко обмінюватись інформацією між клієнтом і сервером.

3 ОПИС АЛГОРИТМІВ

3.1 Загальний алгоритм

1. Авторизація
    1.1. Зчитування даних авторизації
    1.2. Відправка даних на сервер
    1.3. Створення запиту на перевірку прийнятих сервером даних
        1.3.1. Створення SQL запиту
        1.3.2. Відправка SQL запиту
        1.3.3. Вивантаження даних з БД
    1.4. Відправка результату клієнту
2. Відображення сторінки користувача
    2.1. Запит на сервер про для отримання інформації про користувача
    2.2. Відправка даних на сервер
    2.3. Створення запиту на перевірку прийнятих сервером даних
        2.3.1. Створення SQL запиту
        2.3.2. Відправка SQL запиту
        2.3.3. Вивантаження даних з БД
    2.4. Відправка результату клієнту
    2.5. Візуалізація отриманої інформації
3. Відображення сторінки проекту
    3.1. Запит на сервер про для отримання інформації про користувача
    3.2. Відправка даних на сервер
    3.3. Створення запиту на перевірку прийнятих сервером даних
        3.3.1. Створення SQL запиту
        3.3.2. Відправка SQL запиту
        3.3.3. Вивантаження даних з БД
    3.4. Відправка результату клієнту
    3.5. Візуалізація отриманої інформації
4. Чат

4.1. Відправка повідомлення на сервер

4.2. Сервер розсилає текст повідомлення отримувачам

4.3. Візуалізація отриманих повідомлень

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Діаграма класів програмного забезпечення

UML діаграма класів (Рис 4.1).

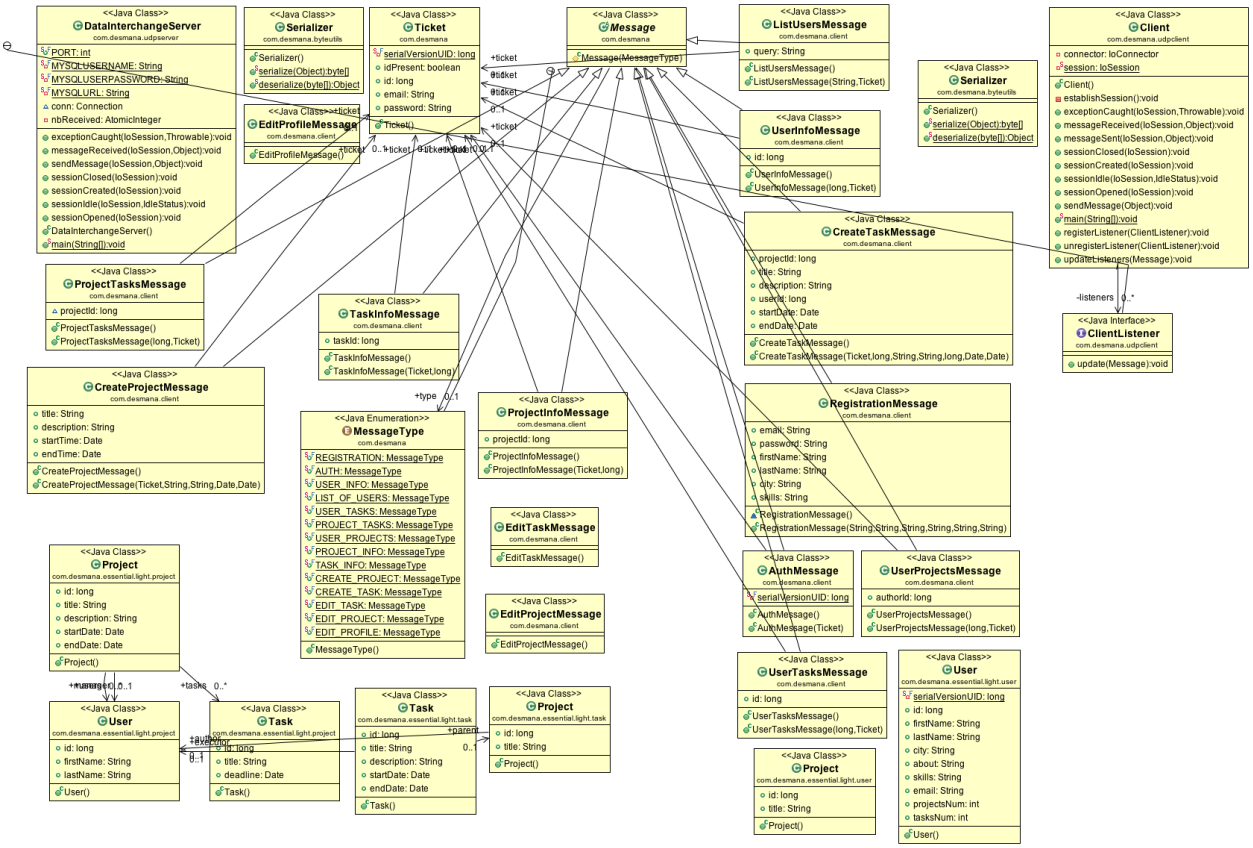

Рис. 4.1 – UML діаграма

4.2 Опис методів частин програмного забезпечення

4.2.1 Стандартні методи

У таблиці 4.2.1 основні стандартні методи програми.

Таблиця 4.2.1 – Стандартні методи

№ п/п	Назва классу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	DataInterchangeServer	sendMessage	Відправка оброблених данних клієнту	IoSession Message	void	Server.java
2	DataInterchangeServer	messageReceived	Отримання запиту з клієнту	IoSession Message	void	Server.java
3	DataInterchangeServer	sessionOpened	Запуск сервера	IoSession	void	Server.java

4.2.2 Користувацькі методи

У таблиці 4.2.2 основні користувацькі методи програми.

Таблиця 4.2.2 – Користувацькі методи

№ п/п	Назва классу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	Client	sendMessage	Відправка данних на сервер	Object	void	Client.java
2	Client	messageReceived	Отримання оброблених данних з серверу	IoSession Object	void	Client.java
3	Client	visualizate	Візуалізація данних на активіті	Message	void	Client.java
4	Client	update	Поновлює інформацію на сторінках програми	Message	void	Client.java

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 План тестування

Складемо план тестування програмного забезпечення, за допомогою якого протестуємо весь основний функціонал та реакцію на виключні ситуації.

a) Тестування коректності введених даних.

1) Тестування при введенні некоректних символів.

2) Тестування при введенні даних зареєстрованого користувача.

b) Тестування авторизації.

1) Тестування при введенні неіснуючих даних.

c) Тестування вікна проекту.

1) Тестування доступу до інформації.

2) Тестування відправки повідомлення.

5.2 Приклади тестування

Проведемо тестування(таблиці 5.1 – 5.5)

Таблиця 5.1 – Приклад роботи програми при введенні некоректних символів

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрите вікно реєстрації
Вхідні дані	E-Mail – andrey2004112gmail.com Password – p@$$80rd First name – Andrii Last name – Mieshkov City – Kyiv Skills – Programming, developer
Схема проведення тесту	Ввести дані та натиснути Register
Очікуваний результат	Повідомлення про некоректність даних
Стан програми після проведення випробувань	Видано помилку «Incorrect e-mail»

Таблиця 5.2 – Приклад роботи програми при введенні даних зареєстрованого користувача

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрите вікно реєстрації
Вхідні дані	E-Mail – andrey2004112@gmail.com Password – p@$$80rd First name – Andrii Last name – Mieshkov City – Kyiv Skills – Programming, developer
Схема проведення тесту	Ввести дані та натиснути Register
Очікуваний результат	Повідомлення про некоректність даних
Стан програми після проведення випробувань	Видано помилку «This e-mail is already registered»

Таблиця 5.3 – Приклад роботи програми при введенні неіснуючих даних

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрите вікно авторизації
Вхідні дані	E-Mail – andrey2004@gmail.com Password – p@$$80rd
Схема проведення тесту	Ввести дані та натиснути Sign in
Очікуваний результат	Повідомлення про некоректність даних
Стан програми після проведення випробувань	Видано помилку «Incorrect e-mail or password»

Таблиця 5.4 – Приклад роботи програми при перевірці даних проекту

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрита власна сторінка
Вхідні дані	-
Схема проведення тесту	Обрати проект та натиснути
Очікуваний результат	Відкриття вікна проекту
Стан програми після проведення випробувань	Відкрито вікно проекту з інформацією

Таблиця 5.5 – Приклад роботи програми при відправці повідомлення

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрита власна сторінка
Вхідні дані	Hello!!!
Схема проведення тесту	Обрати чат та натиснути, ввести та відправити повідомлення
Очікуваний результат	Відправлення повідомлення у чат
Стан програми після проведення випробувань	Відправлено повідомлення у чат

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Робота з програмою

1. Після запуску програми перед користувачем з’являється вікно авторизації з можливістю зареєструватися (Рис. 6.1).

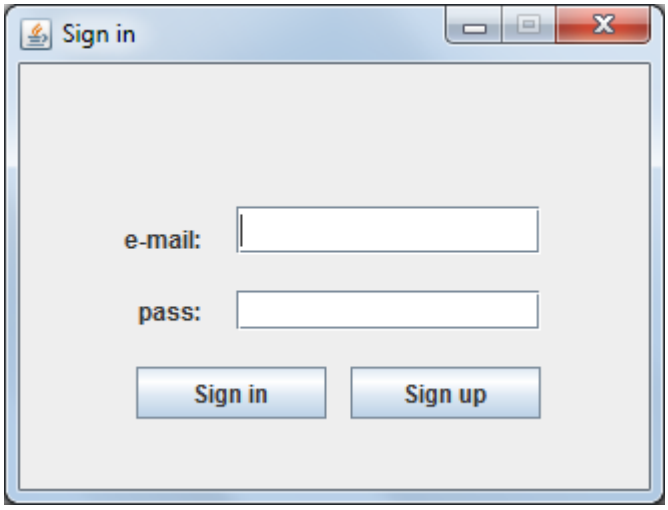

Рис. 6.1 – Вікно авторизації

2. Після успішної авторизації користувач потрапляє на власну сторінку, з якої має можливість перейти до вікна повідомлень або проекту (Рис. 6.2).

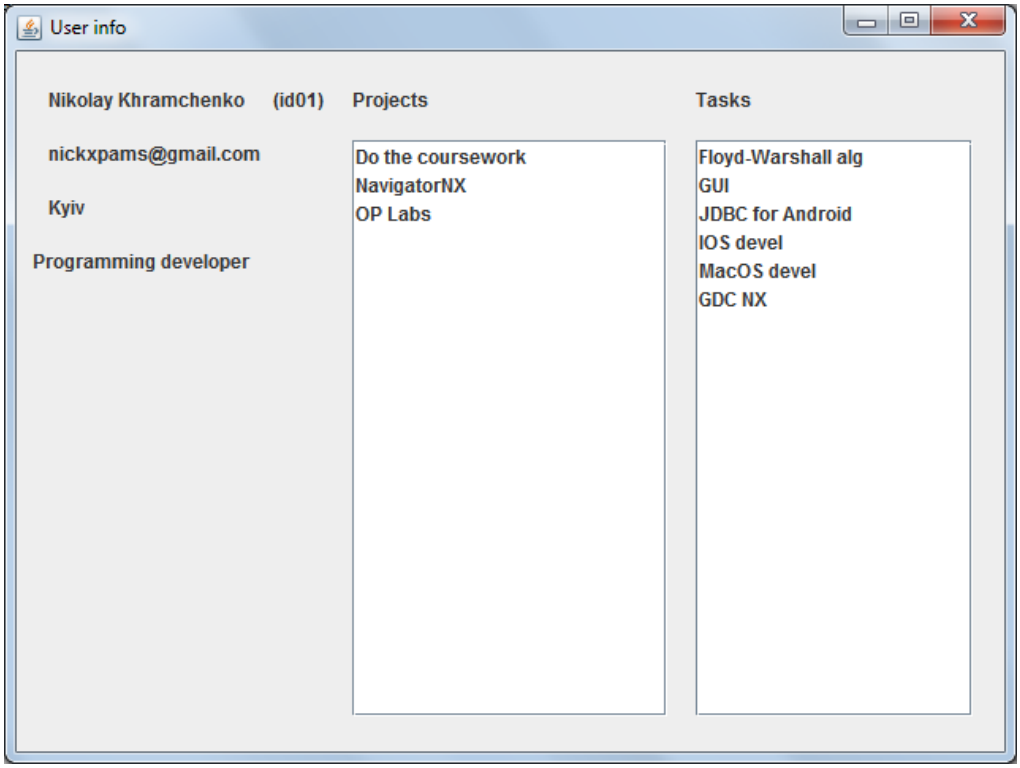

Рис. 6.2 – Вікно сторінки користувача.

3. У вікні проекту користувач має можливість переглянути інформацію про цей проект, завдання проекту, виконавців завдань проекту, строки проекту або перейти до вікна повідомлень (Рис. 6.3).

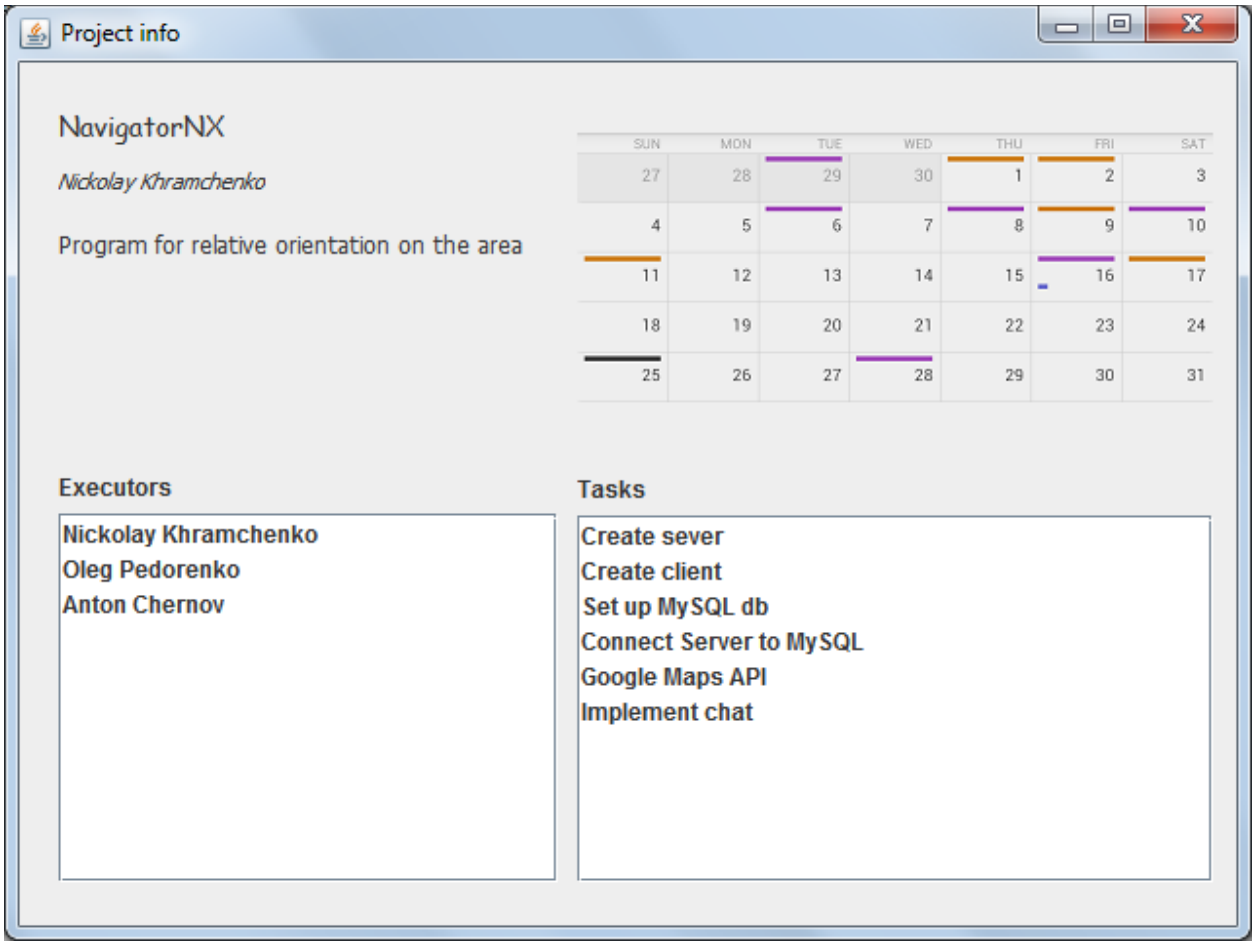

Рис. 6.3 – Вікно проекту

4. У вікні чату користувач має можливість обмінюватися миттєвими повідомленнями з виконавцями проекту (чат проекту) або довільними користувачами (Рис. 6.4).

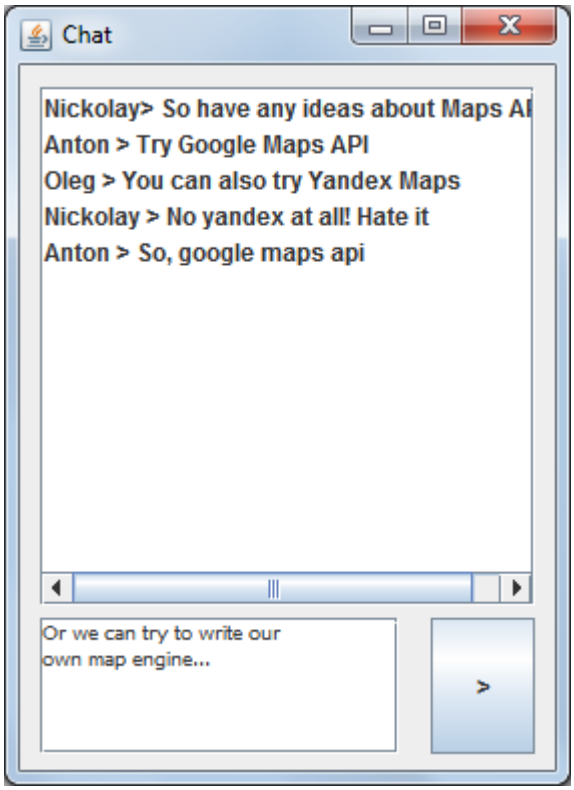

Рис. 6.4 – Вікно чату

5. Для виходу з програми необхідно й достатньо натиснути “X” в верхньому правому куті вікна.

7 АНАЛІЗ РЕЗУЛЬТАТІВ

Таким чином, програма досить проста у використанні і складається з чотирьох вікон. Перше вікно – вікно авторизації, де користувач вводить необхідні данні для аутентифікації, які можна змінити в будь-який момент роботи програми. Друге вікно – де користувач бачить власну сторінку, з якої має можливість виходу у вікно проектів чи повідомлень. У вікні проектів користувач бачить поставлені завдання та строки виконання. У вікні чату користувач має можливість відправляти та отримувати миттєві повідомлення.

ВИСНОВКИ

У ході роботи було проаналізовано існуючі програми, які вирішують проблему взаємної роботи групи користувачів. За результатами аналізу в них було виявлено як переваги так і недоліки. Прикладом недоліків є відсутність мобільної версії додатків або відсутність можливості комунікації між користувачами.

Для вирішення цієї проблеми було розроблено кроссплатформенний клієнт-серверний додаток, який працює за принципом: При запуску програми користувач потрапляє до вікна реєстрації в якому вводить логін та пароль (можливо ввести додаткові данні). При правильному введенні логіна та паролю відкривається вікно власної сторінки. Далі програма синхронізується з сервером і візуалізує вивантажені данні. Є можливість виходу у вікно чату, для обміну повідомленнями, або у вікно проекту.

Клієнтська частина розробленої нами системи може працювати в операційних системах Windows, Linux, MAC OS, Android. Але треба зазначити, що додаток вимагає Інтернет з’єднання. Систему було встановлено на мобільні пристрої, кишенькові та персональні комп’ютери з різними версіями вищезазначених операційних систем та проведено її апробацію.

Після доопрацювання систему можна буде використовувати для голосових дзвінків та розшарування екрану колегам. Надалі є намір додати інтеграцію Google+ та Facebook аккаунтів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mike Loukides. HeadFirst Design Patterns. O'Reilly Media, 2004. ISBN: 0-596-00712-4
2. Eckel Bruce. Thinking Java. U.S. Corporate and Government Sales, 2000. ISBN: 0-13-187248-6
3. Сервіс для розробників програмного забезпечення. URL: http://code.google.com/
4. Вебсайт, створений для публікації новин, аналітичних статей, думок, пов'язаних із інформаційними технологіями, бізнесом та Інтернетом. URL: https://habr.com/
5. Загальнодоступна вільна багатомовна онлайн-енциклопедія. URL: http://wikipedia.org/wiki
6. Сайт питань та відповідей для програмістів. URL: http://stackoverflow.com/

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії

Затвердив:

Керівник Головченко Максим Миколайович

«22» березня 2022 р.

Виконавець:

Студент Мєшков Андрій Ігорович

«22» березня 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання курсової роботи
на тему: "Розробка кроссплатформенного клієнт-серверного додатку для роботи груп користувачів над спільним проектом, а також комунікації між ними"
з дисципліни:
«Основи програмування»

Київ 2022

1. *Мета*: Розробити клієнт-серверний кроссплатформенний додаток для організації роботи над командним проектом групи осіб, а також комунікації між ними.
2. *Дата початку роботи*: «22»березня2022 р.
3. *Дата закінчення роботи*: «14»червня 2022 р.
4. *Вимоги до програмного забезпечення*.
   1) Функціональні вимоги:

      – Функціональним призначенням програми є спрощення роботи груп користувачів над спільною роботою, контролю над поставленими задачами, спільною комунікацією.

      – Програма може експлуатуватися групою розробників програмного забезпечення для контролю виконання поставленої задачі, отримання нових завдань, відображення їх на календарі, а також можливість миттєвої комунікації між членами групи.

   2) Нефункціональні вимоги:

      – Виконуваний код програми має бути написаний на Java;

      – Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

      ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

      ГОСТ 19.106 - 78 - Вимоги до програмної документації.

      ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

5. *Стадії та етапи розробки*:
   1) Об'єктно-орієнтований аналіз предметної області задачі (до 03.05.2022 р.)
   2) Об'єктно-орієнтоване проектування архітектури програмної системи (до 17.05.2022 р.)

3) Розробка програмного забезпечення (до 31.05.2022 р.)
4) Тестування розробленої програми (до 11.06.2022 р.)
5) Розробка пояснювальної записки (до 12.06.2022 р.).
6) Захист курсової роботи (до 19.06.2022 р.).

6. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду розробка кроссплатформенного клієнт-серверного додатку для роботи груп користувачів над спільним проектом, а також комунікації між ними

(Найменування програми (документа))

Електронний носій

(Вид носія даних)

13 арк, 65 Кб

(Обсяг програми (документа), арк., Кб)

студента групи ІП-15 І курсу
Мєшкова А.І.

Код програмного продукту:

Client.java

```
package com.desmana.udpclient;


import com.desmana.Message;
import com.desmana.Ticket;
import com.desmana.byteutils.Serializer;
import com.desmana.client.CreateProjectMessage;
import com.desmana.client.CreateTaskMessage;
import com.desmana.client.RegistrationMessage;
import com.desmana.udpserver.DataInterchangeServer;
import org.apache.mina.core.buffer.IoBuffer;
import org.apache.mina.core.future.ConnectFuture;
import org.apache.mina.core.service.IoConnector;
import org.apache.mina.core.service.IoHandlerAdapter;
import org.apache.mina.core.session.IdleStatus;
import org.apache.mina.core.session.IoSession;
import org.apache.mina.transport.socket.DatagramSessionConfig;
import org.apache.mina.transport.socket.nio.NioDatagramConnector;
import java.net.InetSocketAddress;
import java.util.ArrayList;
import java.util.Date;


/**
 * Client class to interact with server. It is a chain between server and client
 * gui/ui/ux This class provides no processing Just decoding message and
 * updating all listeners
 *
 * @author Olexandr Kovalchuk
 */

public class Client extends IoHandlerAdapter {
	private ArrayList<ClientListener> listeners;
	/** The connector */
	private IoConnector connector;
	/** The session */
	private static IoSession session;
	/**
	 * Create the UdpClient's instance
	 */
	public Client() {
		connector = new NioDatagramConnector();
		connector.setHandler(this);
		// This is used to configure client. I do not think, we need it =)
		// But let this thing be here)
		DatagramSessionConfig dcfg = (DatagramSessionConfig) connector
				.getSessionConfig();
		establishSession();
		listeners = new ArrayList<ClientListener>();

	}
	/**
```

```
 * establishes session
 */
private void establishSession() {
        ConnectFuture connFuture = connector.connect(new InetSocketAddress(
                        "localhost", DataInterchangeServer.PORT));
        connFuture.awaitUninterruptibly();
        session = connFuture.getSession();
}

/**
 * {@inheritDoc}
 */
@Override
public void exceptionCaught(IoSession session, Throwable cause)
                throws Exception {
        cause.printStackTrace();
}

/**
 * {@inheritDoc}
 */
@Override
public void messageReceived(IoSession session, Object message)
                throws Exception {
        IoBuffer data = (IoBuffer) message;
        byte[] buf = new byte[data.limit()];
        data.get(buf);
        Message recreadtedData = (Message) Serializer.deserialize(buf);
        // System.out.println(recreadtedData);
        updateListeners(recreadtedData);
        // TODO: provide message processing in all gui classes
}

/**
 * {@inheritDoc}
 */
@Override
public void messageSent(IoSession session, Object message) throws Exception {
        System.out.println("LOG: session " + session + " sent message "
                        + message);
}



/**
 * {@inheritDoc}
 */
@Override
public void sessionClosed(IoSession session) throws Exception {
        System.out.println("LOG: session " + session + " closed");
}

/**
 * {@inheritDoc}
 */
```

```
@Override
public void sessionCreated(IoSession session) throws Exception {
	System.out.println("LOG: session " + session + " has been created");
}


/**
 * {@inheritDoc}
 */

@Override
public void sessionIdle(IoSession session, IdleStatus status)
		throws Exception {
	System.out.println("LOG: session " + session + " has status " + status);
}



/**
 * {@inheritDoc}
 */

@Override
public void sessionOpened(IoSession session) throws Exception {
	System.out.println("LOG: session " + session + " opened");
}

/**
 * Call this method to send message to server
 *
 * @param message
 *          -- message to send
 */

public void sendMessage(Object message) {
	if (session == null || !session.isConnected()) {
		establishSession();
	}

	try {
		byte[] data = Serializer.serialize(message);
		IoBuffer buffer = IoBuffer.allocate(data.length);
		buffer.put(data);
		buffer.flip();
		session.write(buffer);
	} catch (Exception e) {
		e.printStackTrace();
	}
}

/**
 * TEST The main method : instanciates a client, and send N messages. We
 * sleep between each K messages sent, to avoid the server saturation.
 *
 * @param args
 * @throws Exception
```

```
 */

public static void main(String[] args) throws Exception {
	Client client = new Client();
	CreateTaskMessage cpm = new CreateTaskMessage();
	Ticket t = new Ticket();
	t.id = 3;
	cpm.ticket = t;
	cpm.title = "First";
	cpm.description = "Project";
	cpm.startDate = new Date(2014, 3, 14, 12, 0);
	cpm.endDate = new Date(2014, 6, 11, 13, 50);
	cpm.projectId = 1;
	cpm.userId = 3;
	client.sendMessage(cpm);
	// idk what have to be here
	// we do not need it =)
	// it is here just for test =)
	client.connector.dispose(true);
}



// -----------OBSERVER PATTERN--------------
/**
 * Registers object as client listener
 *
 * @param listener
 *            - listener to register
 */
public void registerListener(ClientListener listener) {
	if (listeners == null) {
		listeners = new ArrayList<ClientListener>();
	}
	listeners.add(listener);
}

/**
 * Stop object being listener
 *
 * @param listener
 *            listener that do not want to be listener anymore
 */

public void unregisterListener(ClientListener listener) {
	if (listeners == null) {
		System.out.println("ERROR: listeners is null");
	} else {
		listeners.remove(listener);
	}
}

/**
 * Call this method each time when the message received
 *
 * @param message
```

```
	 *           message, that have been received
	 */

	public void updateListeners(Message message) {
		for (ClientListener lstnr : listeners) {
			lstnr.update(message);
		}
	}

	/**
	 * This interface must be implemented with all GUI classes, that display the
	 * results
	 */

	public interface ClientListener {
		/**
		 * This method inits listener to update Must have custom implementation
		 * for all listeners
		 *
		 * @param message
		 *           message that have been received
		 */
		public void update(Message message);

	}

	// ----------OBSERVER PATTERN END----------
}
```

Server.java

```
package com.desmana.udpserver;

import com.desmana.Ticket;

import com.desmana.byteutils.Serializer;

import com.desmana.client.AuthMessage;

import com.desmana.client.CreateProjectMessage;

import com.desmana.client.CreateTaskMessage;

import com.desmana.client.ProjectInfoMessage;

import com.desmana.client.RegistrationMessage;

import com.desmana.client.TaskInfoMessage;

import com.desmana.client.UserInfoMessage;

import com.desmana.essential.light.project.Project;

import com.desmana.essential.light.task.Task;

import com.desmana.essential.light.user.User;

import com.mysql.jdbc.Connection;

import com.mysql.jdbc.Statement;

import org.apache.mina.core.buffer.IoBuffer;
```

```
import org.apache.mina.core.service.IoHandlerAdapter;
import org.apache.mina.core.session.IdleStatus;

import org.apache.mina.core.session.IoSession;
import org.apache.mina.transport.socket.DatagramSessionConfig;
import org.apache.mina.transport.socket.nio.NioDatagramAcceptor
import java.io.IOException;
import java.net.InetSocketAddress;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * An UDP server used for performance tests.
 *
 * It does nothing fancy, except receiving the messages, and counting the number
 * of received messages.
 *
 * @author <a href="http://mina.apache.org">Apache MINA Project</a>
 */

public class DataInterchangeServer extends IoHandlerAdapter {
        /** The listening port (check that it's not already in use) */
        public static final int PORT = 18567;
        private static final String MYSQLUSERNAME = "org";
        private static final String MYSQLUSERPASSWORD = "password";
        private static final String MYSQLURL = "jdbc:mysql://192.168.1.221/organizer";
        Connection conn = null;
        /** A counter incremented for every recieved message */
        private AtomicInteger nbReceived = new AtomicInteger(0);
        /**
         * {@inheritDoc}
         */
        @Override
        public void exceptionCaught(IoSession session, Throwable cause)
                        throws Exception {
```

```
            cause.printStackTrace();

            session.close(true);
        }
        /**
         * {@inheritDoc}
         */
        @SuppressWarnings("deprecation")
        @Override
        public void messageReceived(IoSession session, Object message)
                    throws Exception {
            IoBuffer data = (IoBuffer) message;
            byte[] buf = new byte[data.limit()];
            data.get(buf);
            Object recreadtedData = Serializer.deserialize(buf);
            Statement stat = (Statement) conn.createStatement();
            if (recreadtedData instanceof RegistrationMessage) {
                RegistrationMessage rm = (RegistrationMessage) recreadtedData;
                String email = rm.email;
                String password = rm.password;
                String firstName = rm.firstName;
                String lastName = rm.lastName;
                String city = rm.city;
                String skills = rm.skills;
                ResultSet res = stat;
                            .executeQuery("SELECT count(0) FROM users WHERE eMail='"
                                        + email + "';");
                int count = -1;
                if (res.next()) {
                    count = res.getInt(1);
                }
                boolean fl = false;
                if (count == 0) {
                    stat.execute("INSERT into users (eMail, password, fName, lName, city, skils)
"
                                        + "VALUES ('"
                                        + email
                                        + "', '"
```

```
                                        + password
                                        + "', '"
                                        + firstName
                                        + "', '"
                                        + lastName
                                        + "', '"
                                        + city
                                        + "', '" + skills + "');");
                        fl = true;
                }
                stat.cancel();
                stat.close();
                sendMessage(session, new com.desmana.server.RegistrationMessage(fl,
                                "regostrator"));
        } else if (recreadtedData instanceof AuthMessage) {
                AuthMessage am = (AuthMessage) recreadtedData;
                Ticket t = am.ticket;
                long id = t.id;
                String email = t.email;
                String password = t.password;
                String sqlQuery;
                if (!t.idPresent) {
                        sqlQuery = "SELECT count(0) FROM users WHERE id='"
                                        + String.valueOf(id) + "' AND password='" + password
                                        + "';";
                } else {
                        sqlQuery = "SELECT count(0) FROM users WHERE eMail='" + email
                                        + "' AND password='" + password + "';";
                }
                ResultSet res = stat.executeQuery(sqlQuery);
                int count = -1;
                if (res.next()) {
                        count = res.getInt(1);
                }
                stat.cancel();
                stat.close();

                boolean fl = false;
```

```
                fl = count > 0;
                if (fl && !t.idPresent) ;
                        res = stat.executeQuery("SELECT id FROM users WHERE eMail='"
                                        + email + "';");
                        if (res.next()) {
                                t.id = res.getLong(1);
                        }
                }
                com.desmana.server.AuthMessage sam = new com.desmana.server.AuthMessage();
                sam.isSuccessful = fl;
                sam.ticket = t;
                if (!sam.ticket.idPresent && fl) {
                        sam.ticket.idPresent = true;
                }
                sendMessage(session, sam);
        } else if (recreadtedData instanceof UserInfoMessage) {
                UserInfoMessage uim = (UserInfoMessage) recreadtedData;
                long userID = uim.id;
                ResultSet res = stat.executeQuery("SELECT * FROM users WHERE id='
                                + String.valueOf(userID) + "';");
                User user = new User();
                if (res.next()) {
                        user.id = res.getLong(1);
                        user.firstName = res.getString(4);
                        user.lastName = res.getString(5);
                        user.city = res.getString(6);
                        user.skills = res.getString(7);
                        user.projectsNum = 0; // TODO
                        user.tasksNum = 0; // TODO
                        user.email = res.getString(2);
                }
                com.desmana.server.UserInfoMessage suim;
                suim = new com.desmana.server.UserInfoMessage();
                suim.user = user;
                stat.cancel();
                stat.close();
                sendMessage(session, suim);
        } else if (recreadtedData instanceof CreateProjectMessage) {
```

```
                CreateProjectMessage cpm = (CreateProjectMessage) recreadtedData;
                Ticket t = cpm.ticket;
                String title = cpm.title;
                String description = cpm.description;
                Date startDate = cpm.startTime;
                Date endDate = cpm.endTime;
                long mID = t.id;
                stat.execute("INSERT into projects (title, description, manager, startDate, endDate) "
                                + "VALUES ('"
                                + title
                                + "', '"
                                + description
                                + "', '"
                                + String.valueOf(mID)
                                + "', '"
                                + startDate.toString()
                                + "', '" + endDate.toString() + "');");
        stat.cancel();
                stat.close();
                sendMessage(session, new com.desmana.server.CreateProjectMessage());
        } else if (recreadtedData instanceof CreateTaskMessage) {
                CreateTaskMessage ctm = (CreateTaskMessage) recreadtedData;
                String title = ctm.title;
                String description = ctm.description;
                String startDate = ctm.startDate.toString();
                String endDate = ctm.endDate.toString();
                long parent = ctm.projectId;
                long executor = ctm.userId;
                stat.execute("INSERT into tasks (title, description, startDate, endDate, parent,
executor) "
                                + "VALUES ('"
                                + title
                                + "', '"
                                + description
                                + "', '"
                                + startDate
                                + "', '"
                                + endDate
```

```
                    + "', '"
                    + String.valueOf(parent)
                    + "', '"
                    + String.valueOf(executor) + "');");
            stat.cancel();
            stat.close();
            sendMessage(session, new com.desmana.server.CreateTaskMessage());
    } else if (recreadtedData instanceof ProjectInfoMessage) {
            ProjectInfoMessage pim = (ProjectInfoMessage) recreadtedData;
            Project p = new Project();
            ResultSet res = stat.executeQuery("SELECT * FROM projects WHERE id='
                    + String.valueOf(pim.projectId) + "';");
            if(res.next()) {
                    p.id = pim.projectId;
                    p.title = res.getString(2);
                    p.description = res.getString(3);
                    //p.manager = res.getLong(4);
                    p.startDate = new Date(res.getString(5));
                    p.endDate = new Date(res.getString(6));
            }
            stat.cancel();
            stat.close();
            sendMessage(session, p);
    } else if(recreadtedData instanceof TaskInfoMessage) {
            TaskInfoMessage tim = (TaskInfoMessage) recreadtedData;
            Task t = new Task();
            ResultSet res = stat.executeQuery("SELECT * FROM tasks WHERE id='"
                    + String.valueOf(tim.taskId) + "';");

            if(res.next()) {

                    t.id = t.id;
                    t.title = res.getString(2);
                    t.description = res.getString(3);
                    t.startDate = new Date(res.getString(4));
                    t.endDate = new Date(res.getString(5));
                    //t.parent TODO
                    //t.executor TODO
```

```
                }
                stat.cancel();
                stat.close();
                sendMessage(session, t);
        }
        // System.out.println(recreadtedData);
        /*
         * TODO: provide logic to analyse message and send result maybe here
         * should be some method to do that maybe a few methods write to
         * session, using session.write method please serialize output messages
         * to byte[] using Serializer class
         */
}
public void sendMessage(IoSession session, Object message) {
        if (session == null || !session.isConnected()) {
                System.out.println("FAIL");
        }
        try {
                byte[] data = Serializer.serialize(message);
                IoBuffer buffer = IoBuffer.allocate(data.length);
                buffer.put(data);
                buffer.flip();
                session.write(buffer);
        } catch (Exception e) {
                e.printStackTrace();
        }
}

/**
 * {@inheritDoc}
 */
@Override
public void sessionClosed(IoSession session) throws Exception {
        System.out.println("Session closed...");
}
        /**
 * {@inheritDoc}
 */
```

```
@Override
public void sessionCreated(IoSession session) throws Exception {
        System.out.println("Session created...");
}

/**
 * {@inheritDoc}
 */
@Override
public void sessionIdle(IoSession session, IdleStatus status
                throws Exception {
        System.out.println("Session idle...");
}
/**
 * {@inheritDoc}
 */
@Override
public void sessionOpened(IoSession session) throws Exception {
        System.out.println("Session Opened...");
}
/**
 * Create the UDP server
 */
public DataInterchangeServer() throws IOException {
        NioDatagramAcceptor acceptor = new NioDatagramAcceptor();
        acceptor.setHandler(this);
        // The logger, if needed. Commented atm
        // DefaultIoFilterChainBuilder chain = acceptor.getFilterChain();
        // chain.addLast("logger", new LoggingFilter());
        DatagramSessionConfig dcfg = acceptor.getSessionConfig();


        acceptor.bind(new InetSocketAddress(PORT));
        try {
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                conn = (Connection) DriverManager.getConnection(MYSQLURL,
                                MYSQLUSERNAME, MYSQLUSERPASSWORD);
        } catch (InstantiationException e) {
```

```
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    e.printStackTrace();
                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                System.out.println("Server started...");
        }
        /**
         * The entry point. Use it to start server
         */
        public static void main(String[] args) throws IOException {
                new DataInterchangeServer();
        }
}
```