**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

**“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

Кафедра обчислювальної техніки та програмування

**ЗВІТ**

з переддипломної практики

Спеціальність - 123 Комп’ютерна інженерія

Строки проведення практики: 02.09.2020р. – 25.10.2020р.

**Виконав:**

студ. групи КІТ-М119б

Марценюк Б.В. \_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Керівники практики:**

від університету:

Філоненко А.М. \_\_\_\_\_\_\_\_\_\_\_

(підпис)

Харків 2020

ЗМІСТ

[ВСТУП 3](#_Toc54387749)

[1. ШИФРУВАННЯ 4](#_Toc54387750)

[2. ПОНЯТТЯ ВЕБ-СЕРВІСУ ДЛЯ ІНТЕРНЕТ ЛИСТУВАННЯ 5](#_Toc54387751)

[3. ВИБІР МОВИ ПРОГРАМУВАННЯ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ 7](#_Toc54387752)

[4. ОГЛЯД МОВИ ПРОГРАМУВАННЯ JAVA 8](#_Toc54387753)

[5. ОГЛЯД ТЕХНОЛОГІЇ JAVAFX 12](#_Toc54387754)

[6. ОГЛЯД ШАБЛОНУ MVC 14](#_Toc54387755)

[7. АНАЛІЗ І ВИБІР ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ БАЗИ ДАНИХ 16](#_Toc54387756)

[8. ОГЛЯД ШАБЛОНУ DATA ACCESS OBJECT 20](#_Toc54387757)

[ВИСНОВОК 21](#_Toc54387758)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 22](#_Toc54387759)

ВСТУП

Метою переддипломної практики є огляд використовуваних методів та технологій для проектування веб-сервісу інтернет листування. Огляд необхідності шифрування.

Основні задачі проведення переддипломної практики:

* ознайомлення зі структурою підприємства бази практики в цілому та підрозділів, де проходження практики має місце;
* вивчення теоретичних питань, які складають основу задачі, практичне вирішення якої передбачене у виданому студентові індивідуальному завданні;
* виконання пов’язаних з виробничою необхідністю конкретних практичних завдань з спеціальності та у відповідності з програмою практики.

1. ШИФРУВАННЯ

Безпека даних - найважливіше завдання в сучасному світі. Влада, компанії та інші організації втратили багато грошей, а багато інших закрилися, через діяльність хакерів і зловмисників. За минулі роки були розроблені різні схеми шифрування для захисту бази даних від атак зловмисників. Оскільки дані - це життєвий канал кожної організації, існує потреба в безпечному зберіганні даних, щодня генерованих цими організаціями. Хмарне сховище необхідно для віддаленого зберігання даних.

Для багатьох підприємств безпека даних є однією з основних проблем при відправці файлів в хмару. Вони турбуються про те, що їх файли будуть переглянуті або навіть скомпрометовані.

Для захисту цілісності даних, що зберігаються, необхідні методи шифрування даних. У минулому багато компаній відчували себе комфортно, дозволяючи постачальникам хмарних послуг управляти всіма своїми даними, вважаючи, що ризиками безпеки можна керувати за допомогою контрактів, засобів контролю і аудиту. Однак з часом стало очевидно, що постачальники хмарних послуг не можуть виконувати такі зобов'язання.

1. ПОНЯТТЯ ВЕБ-СЕРВІСУ ДЛЯ ІНТЕРНЕТ ЛИСТУВАННЯ

Веб-сервіс для інтернет листування включає у себе декілька понять:

* форуми;
* блоги;
* поштові клієнти;
* месенджери.

Веб-форум або просто форум - інтернет-ресурс, популярний різновид спілкування в інтернеті. На форумі створюються теми для спілкування, що робить його кращим за чат. Всі, кого цікавить певна інформація, можуть зручно й швидко переглянути її на форумі. На форумі є адміністратори (власники форуму) та модератори (обслуговуючий персонал, який стежить за виконанням установлених правил та порядком). Форуми можуть бути присвячені програмному забезпеченню, автомобілям, футбольній команді тощо [1].

Блог - це веб-сайт, головний зміст якого - записи, зображення чи мультимедіа, що регулярно додаються. Для Блогів характерні короткі записи тимчасової значущості [2].

Поштовий клієнт, клієнт електронної пошти, емейл-клієнт - комп'ютерна програма, яка встановлюється на комп'ютері користувача і призначена для одержання, написання, відправлення та зберігання повідомлень електронної пошти одного або декількох користувачів (у випадку, наприклад, кількох облікових записів на одному комп'ютері) або декількох облікових записів одного користувача [3].

Мессенджер - це програма, мобільний додаток або веб-сервіс для миттєвого обміну повідомленнями.

Найчастіше під месенджером розуміють програму, в яку ви пишете повідомлення і де ви їх читаєте. Однак, за кожною такою програмою стоїть мережа обміну повідомленнями, яка теж входить в поняття "мессенджер". Це може бути мережа всередині вашої компанії, а може бути глобальна мережа, наприклад Jabber.

Потрібно сказати, що поняття месенджера вже давно не пов'язують тільки з обміном текстовими повідомленнями. Сучасні месенджери вже стали повноцінними комунікаційними центрами. За їх допомогою можна організовувати спілкування груп учасників, дзвонити, здійснювати відеодзвінки абонентам по всьому світу. Пересилання файлів, фотографій, інтерактивних картинок, наклейок, геолокація - функції, присутні майже у будь-якому месенджері [4].

1. ВИБІР МОВИ ПРОГРАМУВАННЯ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ

Мов програмування, використовуваних для серверної веб-розробки, досить багато: PHP, Ruby, Java, C, Python, Perl і інші.

З технічної точки зору для більшості проектів немає будь-яких обмежень при виборі мови, тобто практично будь-який функціонал сайту або програми може бути успішно реалізований на будь-якому з них, тому призначення мови не накладає ніяких лімітування на проект.

Для вибору розглянуто такі критерії, як наявність об’єкто-орієнтовної парадигми, застосування багатопоточності, багатоплатформеність, статична типізація, швидкість виконання розробленого модуля, зручна робота с БД, оскільки вони є найактуальнішими при виборі мови. Додатковим побажанням є можливість використовувати лише одну мову програмування як для побудування серверної частини, так і для клієнтської частини.

PHP, Rerl, Python, Ruby, звичайно, повністю підходять, як основні мови, проте у випадку що програмний продукт буде використовуватись лише у вигляді веб-сайту. Мова програмування Java надає користувачу створити веб-сервіс інтернет листування на будь-якій платформі, на будь-якій операційній системі, основна частина коду працюватиме будь-де, розробнику необхідно лише створити візуальну частину продукту згідно необхідної платформи. Використовуючи технології JFX можливо створити десктопний додаток, за допомогою технології JSP або JSF продукт легко перетворити у веб-сайт, за допомогою MIDP додаток портується на ОС Android.

Тож враховуючи всі переваги перед іншими мовами, для програмної реалізації було обрано мову програмування Java, оскільки ця мова дозволить швидко та якісно створити готовий продукт.

Програмування на певній мові вимагає наявність відповідного середовища розробки. Для мови Java таким середовищем є САПР IntelliJ IDEA, що призначена для розробки сучасного програмного забезпечення.

1. ОГЛЯД МОВИ ПРОГРАМУВАННЯ JAVA

Мови програмування за класифікацією бувають двох типів:

* мови програмування низького рівня (мова програмування, близька до програмування безпосередньо в машинних кодах використовуваного реального або віртуального процесора;
* мови програмування високого рівня (мова програмування, розроблена для швидкості і зручності використання програмістом. Основна риса високорівневих мов - це абстракція, тобто введення смислових конструкцій, коротко описують такі структури даних і операції над ними, опис яких на машинному коді дуже довгі і складні для розуміння.

На сьогоднішній момент мова Java є однією з найбільш поширених і популярних мов програмування. Перша версія мови з'явилася ще в 1996 році в надрах компанії Sun Microsystems. Java замислювалась як універсальна мова програмування, яку можна застосовувати для різного роду завдань. Поточною версією є Java 12, яка вийшла в березні 2019 року. Java перетворилася з просто універсальної мови в цілу платформу і екосистему, яка об'єднує різні технології, які використовуються в цілого ряду завдань: від створення десктопних додатків до написання великих веб-порталів і сервісів. Крім того, мова Java активно застосовується для створення програмного забезпечення для цілого ряду пристроїв: звичайних ПК, планшетів, смартфонів і мобільних телефонів і навіть побутової техніки. Досить згадати популярність мобільної ОС Android, більшість програм для якої пишуться саме на Java [5].

У створенні мови програмування Java було п'ять початкових цілей:

* синтаксис мови повинен бути «простим, об'єктно-орієнтовним та звичним»;
* реалізація має бути «безвідмовною та безпечною»;
* повинна зберегтися «незалежність від архітектури та переносність»;
* висока продуктивність виконання;
* мова має бути «інтерпретованою, багатонитевою, із динамічним зв'язуванням модулів» [6].

Під «незалежністю від архітектури» мається на увазі те, що програма, написана на мові Java, працюватиме на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції.

Цього можна досягти, компілюючи початковий Java код у байт-код, який є спрощеними машинними командами. Потім програму можна виконати на будь-якій платформі, що має встановлену віртуальну машину Java, яка інтерпретує байткод у код, пристосований до специфіки конкретної операційної системи і процесора. Зараз віртуальні машини Java існують для більшості процесорів і операційних систем.

Стандартні бібліотеки забезпечують загальний спосіб доступу до таких платформозалежних особливостей, як обробка графіки, багатопотоковість та роботу з мережами. У деяких версіях задля збільшення продуктивності JVM байт-код можна компілювати у машинний код до або під час виконання програми.

Основна перевага використання байт-коду — це портативність. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть майже завжди працювати повільніше, ніж скомпільовані у машинний код, і саме тому Java одержала репутацію «повільної» мови. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Одним із таких методів є just-in-time компіляція (JIT), що перетворює байт-код Java у машинний під час першого запуску програми, а потім кешує його. У результаті така програма запускається і виконується швидше, ніж простий інтерпретований код, але ціною додаткових витрат на компіляцію під час виконання. Складніші віртуальні машини також використовують динамічну рекомпіляцію, яка полягає в тому, що віртуальна машина аналізує поведінку запущеної програми й вибірково рекомпілює та оптимізує певні її частини. З використанням динамічної рекомпіляції можна досягти більшого рівня оптимізації, ніж за статичної компіляції, оскільки динамічний компілятор може робити оптимізації на базі знань про довкілля періоду виконання та про завантажені класи. До того ж він може виявляти так звані гарячі точки — частини програми, найчастіше внутрішні цикли, які займають найбільше часу при виконанні. JIT-компіляція та динамічна рекомпіляція збільшує швидкість Java-програм, не втрачаючи при цьому портативності.

Існує ще одна технологія оптимізації байткоду, широко відома як статична компіляція, або компіляція ahead-of-time (AOT). Цей метод передбачає, як і традиційні компілятори, безпосередню компіляцію у машинний код. Це забезпечує хороші показники в порівнянні з інтерпретацією, але за рахунок втрати переносності: скомпільовану таким способом програму можна запустити тільки на одній, цільовій платформі.

Швидкість офіційної віртуальної машини Java значно покращилася з моменту випуску ранніх версій, до того ж, деякі випробування показали, що продуктивність JIT-компіляторів у порівнянні зі звичайними компіляторами у машинний код майже однакова. Проте ефективність компіляторів не завжди свідчить про швидкість виконання скомпільованого коду, тільки ретельне тестування може виявити справжню ефективність у даній системі [7].

Одна з особливостей концепції віртуальної машини полягає в тому, що помилки (виключення) не призводять до повного краху системи. Крім того, існують інструменти, які «приєднуються» до середовища періоду виконання і кожен раз, коли сталося певне виключення, записують інформацію з пам'яті для зневаження програми. Ці інструменти автоматизованої обробки виключень надають основну інформацію щодо виключень в програмах на Java.

Стандартні бібліотеки забезпечують загальний спосіб доступу до таких платформозалежних особливостей, як обробка графіки, багатопотоковість та робота з мережами.

Таким чином можна виділити такі основні можливості мови програмування Java:

* автоматичне управління пам'яттю;
* розширені можливості обробки виняткових ситуацій;
* багатий набір засобів фільтрації введення-виведення;
* набір стандартних колекцій: масив, список, стек і т. д;
* наявність простих засобів створення мережевих додатків (у тому числі з використанням протоколу RMI);
* наявність класів, що дозволяють виконувати HTTP-запити і обробляти відповіді;
* вбудовані в мову засоби створення багатопоточних додатків, які потім були перенести на багато мов (наприклад Python);
* уніфікований доступ до баз даних : на рівні окремих SQL-запитів - на основі JDBC, SQLJ; на рівні концепції об'єктів, що володіють здатністю до зберігання в базі даних - на основі Java Data Objects (англ.) і Java Persistence API;
* підтримка узагальнень;
* підтримка лямбда, замикань, вбудовані можливості функціонального програмування [8].

1. ОГЛЯД ТЕХНОЛОГІЇ JAVAFX

Для того щоб програмний продукт мав певний інтерфейс використаємо технологію JavaFX.

JavaFX дозволяє створювати додатки з багатою насиченою графікою завдяки використанню апаратного прискорення графіки і можливостей GPU.

За допомогою JavaFX можна створювати програми для різних операційних систем: Windows, MacOS, Linux і для самих різних пристроїв: десктопи, смартфони, планшети, вбудовані пристрої, ТВ. Додаток на JavaFX буде працювати скрізь, де встановлена виконуючого середовища Java (JRE).

JavaFX надає великі можливості в порівнянні з рядом інших подібних платформ, зокрема, в порівнянні зі Swing. Це і великий набір елементів управління, і можливості по роботі з мультимедіа, двомірної і тривимірною графікою, декларативний спосіб опису інтерфейсу за допомогою мови розмітки FXML, можливість стилізації інтерфейсу за допомогою CSS, інтеграція зі Swing і багато іншого.

Оскільки бібліотека JavaFX написана як Java API, код програми JavaFX може посилатися на API з будь-якої бібліотеки Java. Наприклад, програми JavaFX можуть використовувати бібліотеки Java API для доступу до власних системних можливостей і підключення до серверних додатків.

Зовнішній вигляд програм JavaFX можна налаштувати. Каскадні таблиці стилів відокремлюють зовнішній вигляд і стиль від реалізації, щоб розробники могли зосередитися на кодуванні. Графічні дизайнери можуть легко налаштувати зовнішній вигляд і стиль програми за допомогою CSS. Якщо у вас є фон веб-дизайну, або якщо ви хочете відокремити користувальницький інтерфейс (UI) і локальну логіку, ви можете розробити аспекти презентації інтерфейсу користувача на мові сценаріїв FXML і використовувати код Java для програми логіка. Якщо ви віддаєте перевагу проектуванню інтерфейсу користувача без написання коду, скористайтеся програмою JavaFX Scene Builder. Під час створення користувальницького інтерфейсу Scene Builder створює розмітку FXML, яку можна перенести на інтегровану середовище розробки (IDE), щоб розробники могли додати бізнес-логіку [9].

Для зручного розподілення коду та орієнтації між моделлю даних, модулем керування та зовнішнім виглядом програми використовується шаблон MVC.

1. ОГЛЯД ШАБЛОНУ MVC

Модель–вигляд–контролер (англ. Model-view-controller, MVC) - архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися [10].

1. АНАЛІЗ І ВИБІР ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Для зберігання інформації користувачів, іх приватних переписок необхідно створити БД.

У колишні часи СУБД займалися збереженням таблиць і інших допоміжних об'єктів в файлах, надавали призначений для користувача інтерфейс для перегляду, редагування таблиць, зміни їх структури, створення і видалення таблиць. Крім того, вони пропонували командну керуючу мову, на якій можна було писати програми, а також давати в командному рядку прості керуючі команди. Такі СУБД були розраховані на однокористувальницькі і однозадачні операційні системи на зразок DOS. Як приклади можна назвати схожі між собою СУБД DBase, FoxPro, Clipper, Paradox.

Залежно від того, з якою базою даних працює СУБД, вона може бути:

* ієрархічною;
* мережевою;
* реляційною;
* об'єктно-реляційною;
* об'єктно-орієнтованою.

В сучасних СУБД застосовується клієнт-серверний підхід. В рамках цього підходу СУБД вдає із себе сервер (постачальник послуг), який приймає мережеві з'єднання, які виходять від програм-клієнтів (споживачів послуг). Клієнтські програми можуть працювати як на тому ж комп'ютері, що і програма-сервер, так і на інших комп'ютерах. Як тільки з'єднання встановлено, клієнт може відправляти запити в сторону сервера, і отримувати від нього відповіді. Таким чином СУБД відповідає тільки за обробку запитів і зберігання даних. Всю інтерактивну частину (тобто пов'язану з взаємодією з споживачем) беруть на себе клієнтські програми.

Клієнт-серверна організація СУБД дає значні переваги:

* централізація зберігання даних. Всі об'єкти баз даних зберігаються на одному комп'ютері або на групі комп'ютерів, спеціально призначених для цього. Ці комп'ютери володіють достатньою швидкодією і обсягом оперативної і постійної пам'яті для виконання покладених на них завдань. Вони можуть перебувати в приміщенні, що охороняється, їх мережевий зв'язок із зовнішнім світом може бути особливо захищений. Ці машини можуть бути підключені до систем безперебійного живлення, так як для багатьох БД дуже важлива надійність в зберіганні даних і безперебійність роботи;
* централізація обробки клієнтських запитів. Оскільки запити від клієнтів можуть надходити в непередбачувані моменти часу і на їх виконання може знадобитися заздалегідь невідомий час, можливі конфлікти запитів. СУБД повинна ставити в чергу на обслуговування такі запити або по можливості обслуговувати їх одночасно. При цьому слід враховувати можливість конфліктних ситуацій, коли, наприклад, кілька запитів намагаються оновити одну і ту ж підмножину записів в таблиці. Або якщо в результаті виконання одного запиту оновлюється підмножина записів, а інший запит вимагає вибірку інформації з цієї ж (частково оновленої) підмножини.

Очевидно, найбільш простий підхід при виборі СУБД заснований на оцінці того, якою мірою існуючі системи задовольняють основним вимогам створюваного проекту інформаційної системи.

Основні критерії вибору СУБД:

* передбачені типи даних;
* реалізація мови запитів;
* мобільність;
* розподіленість;
* засоби проектування;
* підтримувані мови програмування;
* можливість використання в розробці WEB-додатків;
* стабільність виробника;
* поширеність СУБД.

Серед найбільш поширених можна виділити три основні клієнт-серверні СУБД:

* MySQL;
* MS SQL Server;
* Oracle.

MySQL — одна з найпоширеніших систем управління базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. Характеризується високою швидкістю, стійкістю і простотою використання, а також наявністю простої і ефективної системи безпеки. До того ж виділяється підтримкою різних типів таблиць, економним споживанням ресурсів та можливістю синхронізації з іншими базами даних. Ті недоліки, які наразі існують (менша безпечність для великих систем, платна технічна підтримка), не впливають на якість розробки та можуть бути проігнорованими [11].

Oracle – призначена для хмарних середовищ і може бути розміщена на одному або декількох серверах, що дозволяє управляти базами даних, які містять мільярди записів. Деякі з функцій новітньої версії Oracle включають в себе grid framework і використання як фізичних, так і логічних структур.

Це означає, що фізичне управління даними не впливає на доступ до логічних структур. Крім того, безпека в цій версії доведена до найвищого рівня, тому що кожна транзакція ізольована від інших.

Переваги:

* найсвіжіші інновації та вражаючий функціонал вже впроваджені в цьому продукті, оскільки компанія Oracle прагне тримати планку навіть на тлі інших розробників СУБД;
* СУБД від Oracle є вкрай надійною, фактично це еталон надійності серед подібних систем.

Недоліки:

* вартість Oracle може виявитися непомірно високою, особливо для невеликих організацій;

- система може потребувати значних ресурсів вже відразу після установки, тому можливо буде потрібно модернізувати обладнання для впровадження Oracle.

Ідеально підходить для великих організацій, які працюють з величезними базами даних і різноманітними функціями [12].

Оскільки MySQL повністю задовольняє поставленим вимогам, в якості СУБД було обрано саме її.

Для управління базами даних необхідно використовувати певну програму адміністрування. Однією з найпоширеніших є HeidiSQL.

Щоб управляти базою даних з HeidiSQL, користувач має увійти на локальний або віддалений сервер MySQL з прийнятним паролем, створивши сесію. В рамках цієї сесії користувач може управляти базами даних MySQL на сервері MySQL, і від'єднатися після закінчення роботи. Можливості програми цілком достатні для більшості операцій із загальними та просунутими базами даних, таблицями та записами, але розробка залишається у активному стані, щоб забезпечити повну функціональність, котра очікується від фронтенду MySQL.

Можливості та переваги HeidiSQL:

* можливість підключення до декількох серверів в одному вікні;
* можливість підключення до серверу з командного рядка;
* створення та редагування таблиць, даних, збережених процедур, тригерів;
* можливість експорту бази даних в SQL файл для подальшого імпорту в іншу систему.

1. ОГЛЯД ШАБЛОНУ DATA ACCESS OBJECT

Між БД та системою у цілому доцільно створити певний прошарок, який відповідатиме за передачу запитів в БД і обробку отриманих від неї відповідей. DAO абстрагує сутності системи і робить їх відображення на БД, визначає загальні методи використання з'єднання, його отримання, закриття та (або) повернення в Connection Pool.

Об'єкт доступу до даних (DAO) - об'єкт що надає абстрактний інтерфейс до деяких видів баз даних чи механізмів персистентності реалізуючи певні операції без розкриття деталей бази даних. Він надає відображення від програмних викликів до рівня персистентності. Така ізоляція розділює запити до даних в термінах предметної області та їх реалізацію засобами СКБД.

Цей паттерн проектування можна застосовувати до більшості мов програмування, видів програмного забезпечення з потребою персистентності та більшості типів баз даних, але він традиційно асоціюється з застосунками Java EE та реляційними БД доступ до яких здійснюють через JDBC API що пов'язано з походженням паттерна із збірки найкращих практик Sun Microsystems для цієї платформи [13].

Вершиною ієрархії DAO є абстрактний клас або інтерфейс з описом загальних методів, які будуть використовуватися при взаємодії з базою даних. Як правило, це методи пошуку, видалення по ключу, оновлення і т. д.

Реалізація DAO на рівні класу має на увазі використання одного єдиного з’єднання з базою для виклику більш ніж одного методу успадкованого DAO класу. Певні можливості надаються незалежно від того, який механізм зберігання використовується і без необхідності спеціальним чином відповідати цьому механізму зберігання. Цей шаблон проектування застосовується до безлічі мов програмування, більшості програмного забезпечення, що потребує зберігання інформації і до більшої частини баз даних.

ВИСНОВОК

В результаті проходження переддипломної практики отримано відповіді по темі індивідуального завдання, проаналізовано поняття веб-сервіс інтернет листування, проаналізовано мови програмування та обрано основною мову програмування Java, проведено опис технології JavaFX шаблону MVC, було проведено аналіз технологій проектування баз даних та розглянуто шаблон Data Access Object.

Також було поглиблено та закріплено теоретичні та практичні знання, використання яких необхідне при вирішення передбачених програмою практики задач.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* 1. Веб-форум. Доступ до матеріалу: https://bitly.su/Wztw.
  2. Блог це. Доступ до матеріалу: https://uk.wikipedia.org/wiki/D0 %91%D0%BB%D0%BE%D0%B3.
  3. Поштовий клієнт. Доступ до матеріалу: https://ru.bmstu wiki/HeidiSQL.

Обзор мессенджеров. Лучшие и популярные интернет мессенджеры. Доступ до матеріалу: https://www.voipoffice.ru/tags/messendzhery/ .

Введение в Java. Язык программирования Java. Доступ до матеріалу: <https://metanit.com/java/tutorial/1.1.php>.

Design Goals of the JavaTM Programming Language.

Кей С. Хорстманн (2014). Java SE 8. Вводный курс. «Вільямс».

History of Java in Java Application Servers Report 2010 года. TechMetrix Research, 1999.

JavaFX: Getting Started with JavaFX Oracle. Доступ до матеріалу: [https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfxverview. htm](https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfxverview.%20htm)l.

* 1. Krasner, G.E. and S.T. Pope, A cookbook for using the Model-View- Controller user interface paradigm in Smalltalk-80, Journal of Object-Oriented Programming, 1(3), pp. 26-49, August/September 1988, SIGS Publications, New York, NY, USA, 1988.
  2. The mysql Open Source Project on Open Hub: Languages Page — 2006.

Oracle СУБД. Доступно за посиланням: <http://drach.pro/blog/hi-tech/item/145-db-comparison>.

* 1. Core J2EE Patterns - Data Access Objects. Sun Microsystems Inc.