**Лабораторна робота №3**

**Використання мов конструювання**

**програмного забезпечення**

Мета: засвоїти навички використання мов конструювання програмного забезпечення

**Завдання**

1. Обрати та обґрунтувати мову конструювання для розробки ПЗ «Memory.pro».
2. Провести розробку програмного забезпечення.

**Короткі теоретичні відомості**

Мови конструювання включають усі форми комунікацій, за допомогою яких людина може задати рішення проблеми, виконуване на комп’ютері. Як відомо, можна виділити такі типи мов конструювання:

* Конфігураційні мови, що дозволяють задавати параметри виконання програмної системи;

Наприклад: 1С:Предприятие, AutoCad, MathCad

* Інструментальні мови – мова конструювання з елементів. Що використовуються повторно; звичайно будується як сценарна мова (скрипт), виконувана у відповідному середовищі.

Наприклад: FORTRAN; PHP; PERL

* Алгоритмічні

Наприклад: С++, C#, F#, Go, Java, Prolog, Pascal

З'являються більш сучасні методи розробки ПЗ, які дозволяють змінювати код в процесі конструювання.

Невід’ємною частиною конструювання програмного забезпечення є аналіз вимог, необхідних для встановлення методів і засобів їх реалізації.

**Хід роботи**

1. **Вибір та обґрунтування мови конструювання та інших програмних засобів**

ПЗ «Memory.pro» буде розроблено у вигляді web сервісу. Сервіс буде складатись з двох самостійних частин: Memory.pro REST API, та Memory.pro Frontend. Таким чином і мови конструювання та інші програмні засоби діляться на дві групи: для api та для frontend частин.

Для цього є такі причини:

* Незалежність від платформи

На даний момент web – єдиний спосіб розробки ПЗ, що є дійсно кросплатформеним.

Так, також є JavaFx, Electron, тощо. Але той же Electron – це теж web технології, а JavaFx хоча і є кросплатформеною, але це не завжди так. Крім того у наш час Java зазвичай використовується не для графічних додатків.

В той же час будь-який web сервіс можна запустити всюди де є браузер (будь це комп’ютер, ноутбук, планшет чи телефон), при цьому код один і той же. Його не треба компілювати для конкретних систем, тощо.

Крім того, використовуючи такі технології як ServiceWorkers, PWA (progressive web applications) дають можливість «завантажити» веб сервіс, після чого він буде виглядати як звичайний додаток (та навіть, працювати без Інтернету).

* Швидкість розробки

Сучасні web мови та frameworks дозволяють розробляти досить складні додатки за порівняно невеликі терміни.

* Можливості розширення

Завдяки тому, що сервіс розділено на дві частини: API та Frontend при необхідності можна розробити, наприклад, мобільний додаток, використовуючи той самий API.

* 1. Вибір та обґрунтування мови конструювання та інших програмних засобів для “Memory.pro Frontend”

Було вирішено використовувати наступні мови конструювання для Frontend частини сервісу:

* Vue + Vuex
* JavaScript (TypeScript)
* CSS (SCSS)
* HTML (Pug)
* Webpack

Vue – це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня відображення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA, Single-Page Applications).

Vuex – бібліотека для програм Vue що реалізує патерн управління станами. Він служить централізованим сховищем для всіх компонентів програми, з правилами, які гарантують, що стан можна мутувати лише передбачувано.

TypeScript – мова програмування зі статичною типізацією що транслюється у JavaScript і при цьому дуже схожа на JavaScript (завдяки чому код після трансляції дуже схожий на код до трансліції. Завдяки статичній типізації дозволяє створювати більш надійний код, покращує можливості автозавершення.

CSS (SCSS) – Cascading Style Sheets, це мова призначена для стилізації веб сторінок (як і де відображати елементи веб-сторінки), написаних на HTML.

SCSS – препроцесор мови CSS, до додає до мови CSS можливості, яких у неї нема, але при цьому, транслюється у CSS. Додає такі можливості як змінні (вже є і в самому CSS), вкладуваність (скоро буде і в самому CSS), mixins, code splitting, цикли, умовні оператори тощо.

Pug (або Jade) – препроцесор мови HTML та шаблонизатор. Транслюється у HTML, легка у вивчені. Додає такі можливості як: більш наглядний вигляд коду, mixins, code splitting, javascript.

Webpack – сучасні веб додатки є досить великими, та зазвичай вони використовують досить велику кількість додаткових інструментів (SCSS, Pug, TS, icons, image optimization тощо), тому деякі рутинні дії хочеться автоматизувати. Webpack – один з інструментів, що дозволяють це зробити. Webpack – система для розумної збірки JavaScript-коду. Дозволяє розробляти веб додатки для максимальної зручності розробки, а під час збирання, наприклад, перетворює scss в css, pug у html, ts у js, може об’єднувати код у різних файлах в один, та багато іншого.

* 1. Вибір та обґрунтування мови конструювання та інших програмних засобів для “Memory.pro REST API”

Було вирішено використовувати наступні мови конструювання для API частини сервісу:

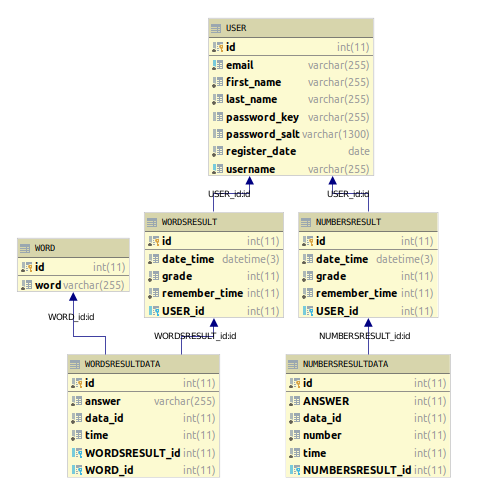
* PHP
* Laravel
* MySQL

PHP – одна з найрозповсюдженіших мов для розробки серверної частити веб додатків. Це досить просто мова, тобто її досить легко «вивчити» (що є одночасно і її плюсом і мінусом). Цю мову часто недолюблюють, через те що код написаний на ній часто може бути досить поганим. Але це лише тому, що мова дозволяє писати по різному. Сучасний PHP код напротив досить надійний. Існує багато фреймворків, що дозволяють одночасно розробляти з великою швидкістю і в той же час створювати надійні веб сервіси.

У якості систему управління базою даних було вирішено використовувати MySql – open-source система управління реляційними базами даних. MySQL є однією з найпопулярніших СУБД і підходить у більшості випадків (в тому числі і для Memory.pro), при цьому може забезпечувати достатні можливості для розширення.

Laravel – безкоштовний open-source PHP фреймворк для створення веб додатків створений Тейлором Отвелом. Призначений для створення веб додатків за архітектурним патерном MVC (model–view–controller). Базується на фреймворку Symfony.

1. **Розробка програмного забезпечення**



AuthService.java

**public class** AuthService {  
 **private** String **loginDataPath** = **"data/config"**;  
  
 **private** UserDao **userDao**;  
 **private** File **loginDataFile**;  
 **private** Optional<LoginData> **loginDataOpt**;  
  
 **public** AuthService(UserDao userDao) {  
 **this**.**userDao** = userDao;  
  
 initLoginDataFile();  
 loadLoginData();  
 }  
  
 **public void** signup(User user) **throws** AuthServiceException {  
 **if** (**userDao**.checkExist(user)) {  
 **throw new** AuthServiceException(**"The user already exist"**);  
 }  
  
 String salt = PasswordUtils.*generateSalt*(512);  
 String key = PasswordUtils.*hashPassword*(user.getPassword(), salt)  
 .orElseThrow(() -> **new** AuthServiceException(**"Internal error"**));  
  
 user.setPasswordSalt(salt);  
 user.setPasswordKey(key);  
 user.setRegisterDate(LocalDate.*now*());  
  
 **try** {  
 **userDao**.add(user);  
 } **catch** (RollbackException e) {  
 **throw new** AuthServiceException(**"Can't save the user"**);  
 }  
 }  
  
 **public void** login(String username, String password) **throws** AuthServiceException {  
 String salt = **userDao**.getSaltByUsername(username)  
 .orElseThrow(() -> **new** AuthServiceException(**"Login failed"**));  
  
 String key = PasswordUtils.*hashPassword*(password, salt)  
 .orElseThrow(() -> **new** AuthServiceException(**"Login failed"**));  
  
 **if** (!checkKey(username, key)) {  
 **throw new** AuthServiceException(**"Login failed"**);  
 }  
  
 setLoginData(username, key);  
 }  
  
 **public void** signupAndLogin(User user) **throws** AuthServiceException {  
 signup(user);  
 login(user.getUsername(), user.getPassword());  
  
 }  
  
 **public boolean** isLoggedIn() {  
 **if** (**loginDataOpt**.isEmpty()) {  
 **return false**;  
 }  
  
 LoginData loginData = **loginDataOpt**.get();  
  
 **boolean** res = checkKey(loginData.getUsername(), loginData.getKey());  
  
 **if** (!res) {  
 removeLoginData();  
 }  
  
 **return** res;  
 }  
  
 **public boolean** logout() {  
 **return this**.removeLoginData();  
 }  
  
 **private boolean** checkKey(String username, String key) {  
 **return userDao**.checkKey(username, key);  
 }  
  
 **private void** initLoginDataFile() {  
 **loginDataFile** = **new** File(**loginDataPath**);  
 **loginDataFile**.getParentFile().mkdirs();  
 }  
  
 **private void** saveLoginData() {  
 **if** (**loginDataOpt**.isEmpty()) {  
 **return**;  
 }  
  
 **try**(FileOutputStream fos = **new** FileOutputStream(**loginDataFile**);  
 ObjectOutputStream out = **new** ObjectOutputStream(fos)  
 ) {  
  
 out.writeObject(**loginDataOpt**.get());  
  
 } **catch** (IOException e) {  
 **return**;  
 }  
 }  
  
 **private void** setLoginData(String login, String key) {  
 **loginDataOpt** = Optional.*of*(**new** LoginData(login, key));  
 saveLoginData();  
 }  
  
 **private boolean** removeLoginData() {  
 **loginDataOpt** = Optional.*empty*();  
 **return loginDataFile**.delete();  
 }  
  
 **private void** loadLoginData() {  
 **try**(FileInputStream fis = **new** FileInputStream(**loginDataFile**);  
 ObjectInputStream ois = **new** ObjectInputStream(fis)  
 ) {  
  
 **loginDataOpt** = Optional.*of*((LoginData) ois.readObject());  
  
 } **catch** (IOException | ClassNotFoundException e) {  
 **loginDataOpt** = Optional.*empty*();  
 }  
 }  
  
 **public** String getLoggedInUsername() {  
 **if** (**loginDataOpt**.isEmpty() || !isLoggedIn()) {  
 **return ""**;  
 }  
  
 **return loginDataOpt**.get().getUsername();  
 }  
  
}

**public class** PasswordUtils {  
 **private static final** SecureRandom ***RAND*** = **new** SecureRandom();  
  
 **private static final int *ITERATIONS*** = 65536;  
 **private static final int *KEY\_LENGTH*** = 512;  
 **private static final** String ***ALGORITHM*** = **"PBKDF2WithHmacSHA512"**;  
  
 **static public** String generateSalt(**int** len) {  
 **if** (len < 1) len = 1;  
  
 **byte**[] salt = **new byte**[len];  
 ***RAND***.nextBytes(salt);  
  
 **return** Base64.*getEncoder*().encodeToString(salt);  
 }  
  
 **static public** Optional<String> hashPassword(String password, String salt) {  
 **if** (salt == **null** || salt.isEmpty()) {  
 **return** Optional.*empty*();  
 }  
  
 **char**[] passwordChars = password.toCharArray();  
 **byte**[] saltBytes = salt.getBytes();  
  
 PBEKeySpec spec = **new** PBEKeySpec(passwordChars, saltBytes, ***ITERATIONS***, ***KEY\_LENGTH***);  
  
 Arrays.*fill*(passwordChars, Character.***MIN\_VALUE***);  
  
 **try** {  
 SecretKeyFactory fac = SecretKeyFactory.*getInstance*(***ALGORITHM***);  
 **byte**[] securePassword = fac.generateSecret(spec).getEncoded();  
 **return** Optional.*of*(Base64.*getEncoder*().encodeToString(securePassword));  
  
 } **catch** (NoSuchAlgorithmException | InvalidKeySpecException ex) {  
 System.***err***.println(**"Exception encountered in hashPassword()"**);  
 **return** Optional.*empty*();  
  
 } **finally** {  
 spec.clearPassword();  
 }  
 }  
  
}

TrainingScreen.java

**public class** TrainingScreen **extends** Screen {  
  
 @FXML **private** BorderPane **fxLeftContainer**;  
 @FXML **private** BorderPane **fxCenterContainer**;  
  
 **private** TrainingSetupComponent **setupComponent**;  
 **private** TrainingMemorizeComponent **memorizeComponent**;  
 **private** TrainingRememberComponent **rememberComponent**;  
 **private** ResultComponent **resultComponent**;  
  
 **private** Parent **setupComponentRoot**;  
 **private** Parent **memorizeComponentRoot**;  
 **private** Parent **rememberComponentRoot**;  
 **private** Parent **resultComponentRoot**;  
  
 **private** TrainingService **trainingService**;  
 **private** ArrayList<String> **trainingData**;  
  
 **private boolean isTrainingInit** = **false**;  
  
 **private** ArrayList<Integer> **timesToMemorize**;  
 **private** ArrayList<String> **answers**;  
 **private int timeToRemember**;  
 **private** String **trainingType**;  
 **private int dataCount**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initChildComponents();  
 }  
  
 @Override  
 **public void** showed() {  
 runSetup();  
 }  
  
 **private void** runSetup() {  
 **fxLeftContainer**.getChildren().clear();  
 **fxLeftContainer**.setCenter(**setupComponentRoot**);  
  
 **setupComponent**.run((trainingType, dataCount) -> {  
 **this**.**trainingType** = trainingType;  
 **this**.**dataCount** = dataCount;  
  
 **if** (**isTrainingInit**) {  
 **var** ans = **alerts**.ask(**"You're already training. Do you want restart it?"**);  
 **if** (!ans) **return**;  
 }  
  
 **isTrainingInit** = **true**;  
  
 runMemorize();  
 });  
 }  
  
 **private void** runMemorize() {  
 initTrainingService();  
  
 **trainingService**.setUp(**dataCount**);  
 **trainingData** = **trainingService**.start();  
  
 **fxCenterContainer**.getChildren().clear();  
 **fxCenterContainer**.setCenter(**memorizeComponentRoot**);  
  
 System.***out***.println(**trainingData**);  
  
 **memorizeComponent**.run(**trainingData**, **trainingType**, (timesToMemorize) -> {  
 **this**.**timesToMemorize** = timesToMemorize;  
  
 runRemember();  
 });  
 }  
  
 **private void** runRemember() {  
 **fxCenterContainer**.getChildren().clear();  
 **fxCenterContainer**.setCenter(**rememberComponentRoot**);  
  
 **rememberComponent**.run(**dataCount**, (timeToRemember, answers) -> {  
 **this**.**timeToRemember** = timeToRemember;  
 **this**.**answers** = answers;  
  
 finishTraining();  
 });  
 }  
  
 **private void** finishTraining() {  
 TrainingResult result = getResult();  
 **int** resultId = **trainingService**.finish(result);  
  
 **fxCenterContainer**.getChildren().clear();  
 **fxCenterContainer**.setCenter(**resultComponentRoot**);  
  
 **resultComponent**.run(**trainingType**, resultId);  
 }  
  
 **private** TrainingResult getResult() {  
 TrainingResult result = **new** TrainingResult();  
  
 result.setTrainingType(**trainingType**);  
 result.setDataCount(**dataCount**);  
 result.setTimesToMemorize(**timesToMemorize**);  
 result.setTimeToRemember(**timeToRemember**);  
 result.setAnswers(**answers**);  
  
 **return** result;  
 }  
  
 **private void** initTrainingService() {  
 User user = getUser();  
  
 **if** (user == **null**) {  
 **alerts**.show(Alerts.*alertErr*, **"Internal Error"**);  
 **return**;  
 }  
  
 **var** em = **common**.getEm();  
  
 **if** (**trainingType**.equals(**"Words"**)) {  
 **trainingService** = **new** WordsTrainingService(user, em);  
  
 } **else if** (**trainingType**.equals(**"Numbers"**)) {  
 **trainingService** = **new** NumberTrainingService(user, em);  
  
 } **else** {  
 **alerts**.show(Alerts.*alertErr*, **"Wrong Training Type"**);  
 System.*exit*(1);  
 }  
  
 }  
  
 **private void** initChildComponents() {  
 **try** {  
 **var** setupPair = loadComponent(**"setup"**);  
 **var** memorizePair = loadComponent(**"memorize"**);  
 **var** rememberPair = loadComponent(**"remember"**);  
 **var** resultPair = loadComponent(**"result"**);  
  
 **setupComponent** = (TrainingSetupComponent) setupPair.getValue();  
 **memorizeComponent** = (TrainingMemorizeComponent) memorizePair.getValue();  
 **rememberComponent** = (TrainingRememberComponent) rememberPair.getValue();  
 **resultComponent** = (ResultComponent) resultPair.getValue();  
  
 **setupComponentRoot** = setupPair.getKey();  
 **memorizeComponentRoot** = memorizePair.getKey();  
 **rememberComponentRoot** = rememberPair.getKey();  
 **resultComponentRoot** = resultPair.getKey();  
  
 } **catch** (ComponentException e) {  
 **alerts**.show(Alerts.*alertErr*, **"Cant'l load Training Components"**);  
 e.printStackTrace();  
 }  
 }

**Висновок:** виконавши лабораторну роботу, я обрав та обґрунтував на основі вимог до ПЗ мову конструювання та інші програмні засоби; засвоїв навички використання мов конструювання програмного забезпечення.