НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

## МАГИСТЪРСКИ ФАКУЛТЕТ

# ДЕПАРТАМЕНТ "ИНФОРМАТИКА"

**ПРОГРАМА „Софтуерни технологии в Интернет”**

**I Курс, 2 семестър**



### КУРСОВА ЗАДАЧА

**КУРС: INFM263 Проект: Програмиране на .Net и J2EE**

**ТЕМА: Клауд базирано решение за мениджмънт на портфолио от криптовалути**

**НА СТУДЕНТА** : Георги Иванов Георгиев

**Фак. №:** f88683

**Дата:** 22/05/2018 г.

**Преподавател:**

/Илиян Занкински/ …………………

София

1. **Цел и задачи на проекта**

Проекта има за цел да създаде базова версия на Cloud приложение със Single page уеб клиент, за улеснено управление на портфолио от виртуални валути. Основната задача, която седи пред проекта е той да бъде максимално удобен за потребителя, лесен за ползване и да дистрибутира ресурсите необходими за изпълнението на калкулационни задачи, както и достъпа до клиентските данни между клауд Worker и Web роли.

1. **Структура, съдържание и workflow**

Проектът се състои от 4 основни страници:

* Начална страница – цели да запознае потребителя със същността на платформата и да изтъкне нейните преимущества
* Register страница – позволява регистрация на нови потребители
* Login страница – позволява аутентикация на потребители със съществуващи акаунти
* Portfolio страница – позволява добавяне на количества от различни по вид криптовалути, както и следене на тяхното развитие (курс и сума на портфолиото)

За пълноценното използване на платформата от потребителя се изисква задължителна регистрация. За успешното регистриране на потребител е необходимо да бъдат въведени следните задължителни данни:

* Име (Name) – пълното име или псевдоним на потребителя. Използва се за обръщение към него в страницата Portfolio;
* Имейл (E-mail) – уникален имейл използван като ключ за ново регистриран потребител. Регистрираните имейли в платформата не могат да се повтарят.
* Парола (Password) – използва се за аутентикация. Полето има следните изисквания – поне 8 символа; поне 1 малка буква; поне 1 главна буква; поне 1 число; поне 1 специален символ
* Повтаряне на паролата (Repeat password) – използва се за удобство на потребителя. Осигурява някакво ниво на сигурност, че потребителят не е сбъркал първоначалното въвеждане на парола, като се изисква от него да го повтори.

След регистриране, потребителят е пренасочен към страницата за аутентикиране. Там той трябва да въведе потребителски имейл и парола, които се сравняват с тези записани в SQL база данни. По този начин удостоверява пред приложението, че именно той е собственикът на данните към даден акаунт. Ако потребителят не въведе правилна комбинация от потребителско име и парола той бива уведомен, че въведените данни са грешни. Ако данните са коректни потребителят се пренасочва към страница Portfolio. Тук той следва да въведе различните позиции(комбинация от вид криптовалута и брой притежавани единици) от своето портфолио. За целта избира бутонът въведи маркиран с иконата плюс. Това действие отваря контекстова страница с лист от криптовалути (комбинация от име и позиция според пазарната капитализация на валутата) попълнен в реално време чрез платформата coinmarketcap и филтър позволяващ филтриране по име. Избирането на желаната валута отваря нова контекстова страница позволяваща въвеждането на размер от желаната валута и прибавянето ѝ към потребителското портфолио. Страницата Portfolio съдържа още три възможности за потребителя – изтриване на цялото портфолио, обновяване на информацията за размерът на портфолиото и цените на отделните криптовалути и сваляне на файл в текстов формат съдържащ цялата информация за портфолиото на конктретния потребител.

1. **Използвани технологии**

Проектът е от тип Azure Cloud Service, а използваният програмен код е C#. Проектът се развива и функционира в клауд средата на Microsoft – Azure, и се възползва от основните компоненти на средата (Compute и Storage), за да функционира правилно. За целта са изградени 2 основни и 1 спомагателни проекта:

* **WebRole структура**

Основен движещ механизъм в архитектурата на приложението е ASP .NET MVC 5. Фреймурка позволява работа на Single page application-а в Microsoft Azure среда. Чрез него са реализирани

* Навигация в рамките на основните модули от приложението
* Комуникация със SQL сървър
* Частична автентикация на потребители
* Комуникация с Worker ролята, като в основата на тази комуникация стои нуждата от изпълнение на дадена изчислителна или комуникационна задача, чиито резултат може да бъде асинхронен

Приложението е от тип Single page application. Потребителят се намира в една единствена уеб страница, а навигирането в нея става посредством подмяна на структурата на DOM дървото. Функционалността на проекта е изградена чрез javascript компоненти. Използваният синтаксис е Ecma Script 6 и HTML 5.

В своята основа приложението разчита на фреймуъркa с отворен код Vue.js. Причината да предпочета използването на този сравнително нов фреймуърк са неговите силни страни, а именно:

* + Малък и олекотен размер
  + Лесното разбиране на същността му и бързото навлизане с цел разработка на приложения
  + Лесната интеграция
  + Добрата документация
  + Възможността му да се комбинира с Ecma Script 6 синтаксис
  + Поддръжката на двупосочна комуникация между компонентите

За реализирането на проекта бе необходимо да използвам следните допълнения към основния фреймуърк:

1. Lodash – библиотека за операции върху масиви
2. Vue-router – библиотека осигуряваща routing в рамките на приложението
3. Vuetify – CSS фреймуърк базиран на т.нар. material дизайн, набиращ все по-голяма популярност. Преимуществата му са минималистичен и функционален responsive дизайн. Фреймуърка използва flex box контейнери за изграждането на рамката на дадена страница.
4. Vue-resource – библиотека за осъществяване на AJAX заявки към API-та на трети страни (в контекста на приложението това е <https://coinmarketcap.com/api/>). Библиотеката обвива функционалностите предоставяни от стандартния XMLHttpRequest.

* **WorkerRole структура**

В своята основа проектът представлява прост изпълним C# проект. Негова основна задача е изпълнението на изчислителни и записващи в базата операции. Проекта комуникира с WorkerRole, чрез съобщения в т.нар. клауд опашка. След изпълнение на своята задача (обновяване цените на валутите в порфолиото на конкретен потребител) проекта пише в базата новите промени и нотифицира клиента.

* **PortfolioCommon**

Проект от тип C# Class Library предоставя обща функционалност за писане в паметта на клауд приложението (Blob, Queue Access), функционалност за писане в SQL Server база данни, модели на основните обекти използвани от системата (UserEntity, CoinEntity), мениджър с конкретна имплементация за изпълняване на методи върху Blob, Cloud и SQL data access, и файл с константи.

1. **Програмна логика и връзки между компонентите на cloud приложението**

Предоставеният custom код се състои най-общо от views и components Javascript файлове. View файловете съдържат в себе си голяма част от презентационната и бизнес логика на отделните страници. За разлика от тях component файловете съдържат преизползваема логика и презентация, която е подходяща за повече от едно View или е обща за много на брой View-та. Съществува и трета група javascript файлове – config файлове. Те служат за настройка на средата на Vue приложението и комуникацията с т.нар. notification hub – част от signal R комуникацията между сървър и клиент.

Съхранението и достъпът до данните е реализиран чрез комуникация между фронт и бек енд осъществена чрез ajax заявки. Те предоставят на разработчика достъп до таблиците User, Portfolio и Coin, които са част от SQL база данни.

Prototypes.js е специфичен файл надграждащ базовите обекти предоставяни от javascript с необходима логика за изпълнението на приложението. Стартирането на сайта започва при достъпване на определен localhost порт. MVC приложението сервира своя Home/Index.cshtml, който съдържа скелета на приложението, главният portfolio-app елемент, и файл app.js, който създава т.нар. application компонент, в чиито контекст се развиват и разгръщат потребителските действия.

В рамките на приложението са реализирани следните фукнционалности:

* Регистрация и аутентикация – функционалността е реализирана посредством комуникация между фронт и бек енда. Регистрацията попълва User table, а аутентикацията сравнява данните въведени по време на Login с тези реално записани в таблицата.
* Портфолио менижмънт – функционалността е реализирана чрез писане по ключ към Portfolio таблицата. Потребителят записва референция (вторичек ключ) към даден Coin - обекти, съдържащи стойност, id, размер, цена и др.
* Валидация на данните – функционалността е реализирана чрез стандартни vuetify компоненти, използващи функции за Regular expression, data comparison и др.
* Live data – функционалността е реализирана чрез извършването на Ajax заявки към външно за приложението API, изчакването на резултата от заявката чрез Promise и обработката на данните по подходящ за бизнес логиката на приложението начин.
* Обновяване на портфолио – функционалността е реализирана чрез изпращане на съобщение по клауд опашка към WorkerRole. WorkerRole проекта приема съобщението, намира портфолиото на потребителя, взима неговите валути и обновява цените чрез комуникация към coinmarketcap open api. След приключване клиента се нотифицира и неговата страница е се обновява с новите цени, както и новопресметната обща стойност на портфолиото.
* Сваляне на портфолио – функционалността е реализирана чрез записване на стринг в блоб. След изпращане на заявка за сваляне на портфолиото проекта изтегля записания стринг по ключ имейл на юзъра и изпраща съдържанието му като файл на потребителската машина.