Университетска система за споделяне на информация

# Глава 1

В тази глава ще бъде разгледан основния проблем при споделяне на информацията, ще бъдат очертани целите и задачите на настоящата дипломна работа и ще бъде посочена структурата на дипломния проект.

## Актуалност на проблема

21 век, век на технологичен напредък и развитие, век на споделяне на информация, век на информационните системи, век на развитие за океана, наречен Интернет.

Споделянето на информация е в основата на ежедневието ни днес. Използването на Интернет, търсенето на информация заема огромна част от живота ни. Толкова голяма част, че вече не можем да си представим как бихме се справили с ежедневните задачи без интернет и Google.

Ако се опитаме да очертаем границите на споделянето на информация и доколкото присъства в живота ни, то може да започнем със споделянето на информация в работна среда и как основната задача днес се състои в постоянно събиране и обработване на данни и визуализирането им в подходяща форма. Другата граница стига до най-простичките неща от живота, а именно ежедневно осведомяване с новини на локално и глобално ниво, мигновено търсене на всякакъв вид информация, както и споделяне на информация с близките.

Ние буквално сме заринати от информация, разпръсната навсякъде из мрежата. Развитието на информационните системи и технологии ни позволява навсякъде и по всяко време да имаме достъп до информация на един клик разстояние. Ние като съвременни интернет потребители имаме достъп до милиони източници на информация.

Но наличието на тонове и тонове информация в мрежата се оказва нож с две остриета.

Ползите са безспорни.

Аз като студент, изучаващ именно информационните системи се сблъсках с другия край на острието, а именно трудното откриване на нужната за мен информация, разпръсната в стотици сайтове, тетрадки на институции и т.н.

Съществуват добри модели на платформи за специфични обекти на дадена предметна област, които позволяват обмен на информация между заинтересованите лица и откриване на информация само по няколко зададени критерий, без нужда от продължителни проучвания в Интернет и преглеждане на множество източници.

Всеки студент се интересува от неща свързани с университета като например стипендии, за които може да кандидатства, стажове, отменени лекции, упражнения и т.н. Понякога самите преподаватели имат нужда да обявят информация, било то за специална стипендия, работа или нужда от помощ.

Всеки завършил студент се интересува от възможностите за работа, свързани с образованието, което е получил. Цялата тази информация днес ние получаваме благодарение на социалните мрежи като Facebook, Instagram, Youtube, официалните сайтове на компании и организации.

Голяма част от преподавателите за съжаление не членуват във групите в социалните мрежи и нямат достъп до основния източник на информация за студентите.

Моята студентска практика показва, че ако искам да съм информирана с цялата информация, касаеща студентския ми живот, трябва ежедневно да следя всички социални мрежи, сайтовете на софтуерни компании, търсачки за работа, официалния сайт на университета, официалния сайт на факултета и т.н. Повечето от нас нямат време за това.

Избираме си 2 – 3 източника на информация, които следим по-редовно и не обръщаме внимание на останалите или им обръщаме много рядко.

За съжаление другият недостатък на разпръснатат информация е че е доста неразбираема. Т.е не само,че трябва да проверим 100 страници и мрежи за определена информация, която ни интересува, а после трябва да и да се опитваме да разгадаем написаното.

Пример от личната ми практика е търсенето на информация за определена стипендия: информацията или липсва изцяло и аз като студент, имащ право на стипендия нямам право на информация. Или другата крайност – попадам на обява за стипендия, която е разпръсната в сайта на университета, останалата половина е в сайта на Министерство на образованието и наука и още няколко съпътстващи ги. Като резултат имаме изгубени часове и неясното дали имам право на тази стипендия. Според статистика на Google средният брой търсене думата “стипендия” за последните 4 часа е над 80 пъти.

По този начин се оказва, че във векът на споделянето на информация, с наличието на широка гама от социални мрежи, сайтове, устройства за нотификации, смартфони, таблети и часовници ние сме изгубени в хаоса и оставаме неосведомени за важните неща.

## Цел и задачи

Целта на настоящата дипломна работа е да се изтъкнат проблемите със споделянето и откриването на информация, свързана с университета, да се очертае и систематизира в категории, най-важната, за студентите информация, за да бъде достъпна на един клик разстояние.

Решаването на тези проблеми ще бъде практически демонстрирано със създаване на прототип на университетска система за споделяне на информация.

Задачите на дипломната работа ще бъдат:

1. Идентифициране на процесът по споделянето на информация в университета - тук ще разгледаме някои основни дефиниции от предметната област, текущи методи за споделяне на университетска информация
2. Анализ на проблемите на текущото състояние - откриване на недостатъците на текущите методи за споделяне и намиране на информация, свързана с университета. Ще направим изводи, на базата на които ще формулираме идея за решаване на текущите недостатъци.
3. Проектиране на софтуерна система за споделяне на информация - задачата тук е да се уеб базирана система, която обединява функционалностите на текущите системи за споделяне на информация, като поправи недостатъците им.
4. Изследване на технологии за реализация на системата и провеждане на сравнителен анализ за предимствата и недостатъците им. На база получените резултати ще бъдат избрани най-подходящите технологии за имплементацията на системата.
5. Реализация на системата - информационната система ще бъде имплементирана и представена

## Очаквани ползи

Основното предимство, което ще бъде достигнато с информационната система, която ще бъде реализирана в рамките на текущата дипломна работа е осъществяване на споделянето на информация между членовете на една университетска организация. Тя ще донесе следните ползи за заинтересованите лица:

1. Студентите, преподавателите или собственици на софтуерни компании ще могат да преглеждат обяви, свързани с университетския живот. Обявите ще бъдат в 3 категории: работа, стипендия и благотворителност

2. Студентите, преподавателите или собственици на софтуерни компании ще могат да публикуват обяви, които касаят студентите от Софийски Университет “Св. Климент Охридски”, да описват своята обява в съответната категория, да я редактират или изтрият.

3. Всеки регистриран потребител ще има възможност да прегледа обявите, които системата предполага,че ще представляват интерес за него. Препоръчването на обяви на потребителя става със специално създадена логика.

4. Системата е предназначена и за улеснение на служителите в кариерен център към Софийски Университет “Св. Климент Охридски”. Те могат да изпълняват ролята на администратори, които поддържат точността на информацията, която се съхранява в системата, а именно информация за обяви и акаунти на потребители. Администраторите имат същите права, като всеки регистриран потребител.

5. Дефиниране на цели към бъдещи трудове - в рамките на настоящата дипломна работа уеб системата ще бъде реализирана за поддръжка на обяви в 3 категории: работа, стипендия и благотворителност. Системата ще бъде проектирана и реализирана по гъвкав начин за лесното и разширение с бъдещи категории.

## Структура на дипломната работа

Глава 1. Увод - в тази глава ще разгледаме актуалност на проблема, целта и задачите на настоящата дипломна работа, очакваните ползи от реализацията, както и ще очертаем структурата на дипломната работа

Глава 2. Преглед на споделянето на информация в университет СУ”Св.Климент Охридски”- в тази глава ще разгледаме някои основни дефиниции от предметната област, съществуващи методи за споделяне на информацията. Ще анализираме проблемите, ще изработим критерий за сравнение между отделните решения и ще направим сравнителен анализ по създадения критерий.

Глава 3. Използвани технологии, платформи и/или методологии- в тази глава ще представим различните технологии, които могат да бъдат използвани за решаване на проблемите очертани във втора глава. Чрез сравнителен анализ ще изберем най-подходящите, ще представим техните предимства и недостатъци.

Глава 4. Анализ- в тази глава ще бъдат представени концептуални модели, ще бъдат изяснени потребителските изисквания, ще се оформи ясна визия на проекта, ще бъде представен модел на потребителските случаи, ще бъдат очертани нефункционалните изисквания.

Ще се направи примерно планиране по методологията Scrum.

Глава 5. Проектиране **-** в тази глава ще бъде представена общата архитектура на системата, ще бъдат представени слоеве, модули, компоненти. Ще бъде представен модел на данните, файлова структура на проекта, диаграми на дейностите и състоянията.

Ще бъдат представен Потребителски интерфейс.

Глава 6. Реализация, тестване/експерименти- в тази глава ще бъде описано как системата е реализирана. Ще бъдат представени части от кода, ще се дефинира набор от функционални системни тестове и ще бъде анализиран резултатът от изпълнението им.

Глава 7. Заключение **-**ще се направи обобщение на осъществената дипломна работа и ще се отбележи кои от първоначално заложените цели са били постигнати.

Накрая ще се прикачат приложения, като терминологичен речник и използвана литература.

# Глава 2

В тази глава ще разгледаме някои основни дефиниции от предметната област и съществуващи методи за споделяне на информацията. Ще анализираме проблемите и ще създадем критерий за ефикасността на съществуващите решения.

## Основни дефиниции

„Система“ ще наричаме съвкупността от софтуерните модули и компоненти, които са проектирани и разработени в настоящата дипломна работа. Всеки компонент изпълнява дадена функционалност.

“Уеб базирана система” - софтуерна система с единна база от данни и централизирана архитектура, която позволява съхранението на всички данни на едно място. По този начин всеки от потребителите на системата има право да създава обяви, видими за останалите потребители на системата. Тя може да бъде бъдат използвана виртуално, на всякаква платформа, с помощта на уеб браузър.

„Клиент-сървър“ е тип мрежова архитектура, която отделя клиента от сървъра. Моделът клиент/сървър е основата на днешните разпределени системи. Той е отговор на ограниченията типични за традиционния модел на централизирани ресурси, където главният компютър предоставя достъп до бази от данни. Архитектурата клиент/сървър има три основни компонента: клиент, сървър и връзката помежду им.

Клиентът е програмата, очакваща да получи услуга от друга програма. Той взаимодейства с потребителя чрез клавиатурата, дисплея или друго входно/изходно устройство. Клиентът няма директни отговорности към достъпа до данни. Той само изпраща заявки до сървъра и показва върнатите резултати на екрана.

Сървърът е системата, която предоставя набор от услуги на клиента. Той има за задача да приема и обработва клиентските заявки и да връща отговора обратно.

“Web API” - Уеб API е приложно-програмен интерфейс, предназначен за уеб сървър или уеб браузър. Идеята е да се предоставят програмни интерфейси на група от услуги, които да се използват от различни приложения. Когато се използва в контекста на уеб програмиране, едно API е дефинирано като група от HTTP извикващи съобщения, заедно с дефиниция на структурата на отговарящите съобщения, което обикновено е JavaScript Object Notation (JSON) формат.

ORM (Object-relational mapping) - програмна техника за превръщане на данните от от несъвместим за съответна система тип, към съвместим. Например изграждане на класове модели, съответстващи на обекти от базата от данни.

ФСС - Факултетният студентски съвет при ФМИ е организация, представляваща студентите и докторантите във Факултета по математика и информатика към СУ „Св. Климент Охридски“. Основната цел на ФСС е да представлява студентската общност във ФМИ към СУ. Основна задача на ФСС е да информира учащите чрез подходящи информационни канали за основни въпроси, които ги касаят.

## Съществуващи проблеми

На голяма част от студентите им се случва да научават колеги или преподаватели за съществуването на определена стипендия и след това да не могат да намерят достатъчна информация за нея. Или базирайки се на идеята, че само всичко, което го има публикувано на официалния сайт на университета е валидно за студентите от университета, веднага предполагат, че нямат право да кандидатстват за дадената стипендия. По този начин поне 60% от студентите изобщо не кандидатстват за стипендия и не използват правата си да бъдат одобрени.

Съществува и друг сценарий, в който студентите научават за съществуването на дадена стипендия от официалния сайт на университета, но информацията в него е недостатъчна. На студента му се налага да потърси за тази стипендия в още няколко сайта, докато установи все още ли е активна, срокове за кандидатстване и т.н.

Най-лошият сценарий е съществуването на стипендии, които не се оповестяват в интернет пространството. Например държавните стипендии, за които може да се кандидатства всяка година. За тях няма почти никаква информация в интернет, като по този начин често студентите пропускат сроковете за кандидатстване

Често преподавателите като част от различни световни организации получават правото техни студенти да кандидатстват за различни стипендии или платени обучения. Преподавателите успяват да разкажат за тази възможност използвайки някои от социалните мрежи или оповестяват на лекция само на присъстващите студенти.

Актуален проблем, на който се натъкнах разучавайки начините за споделяне на информация свързани с университета е липсата на място, където преподаватели или студенти могат да потърсят помощ свързана с благотворителност. Темата за благотворителност присъства в речника на Факултетния Студентски Съвет от известно време. За съжаление студентите и преподавателите трудно достигат до техните обяви.

Съществува сценарии, в който член на университета се нуждае от помощ, свързано с благотворителност, а където това да бъде публикувано.

Друг основен проблем е свързан с желанието на софтуерните компании да подбират служители именно от Факултета по Математика и Информатика към СУ ”Св. Климент Охридски”. Процесът по, който трябва да минат специалистите по набиране на кадри от съответната софтуерна компания е дълъг и тежък, а обявата им се появява или на хартия, разлепена в университета или в някоя социална мрежа. По този начин не получават почти никаква публичност, съответно броят на кандидатстващите е изключително малък.

## Съществуващи решения

Текущите решения за справяне с горните проблеми са много на брой, но повечето от тях неефективни и неработещи за съответната ситуация.

Обяви за стипендии, както за стажове и работа се публикуват в официалния сайт на университета в специална секция наречена - “Кариерен център”. В нея се публикуват обяви само от служител към кариерния център. Т.е ако студент или преподавател разберат за съответна стипендия, те не могат да публикуват обява, а трябва задължително да се свържат със служител. Малко то студентите знаят за съществуването на тази секция в официалния сайт на университета.

Същият е сценарият и с публикуване на обяви за работа от дадена софтуерна организация - тя трябва задължително да се свърже със служител за да стане обявата публична.

Т.к административните служители знаят, че секцията “Кариерен център” е рядко използвана, тяхната работа включва и публикуване на същите обяви в социалните мрежи - Facebook групи, Instagram реклами, YouTube клипчета.

Именно социалните мрежи са и начинът да се разпространи информация от студент до останалите студенти или от преподавател към останалите студенти и преподаватели.

Тук изниква обаче проблемът всички социални мрежи ежедневно ли трябва да се проверяват, за да не пропуснем срока за дадена стипендия?

Колко достоверна е информацията публикувана в социалните мрежи?

Какво се случва с академичните лица, които нямат акаунти в социалните мрежи?

Друг подход на служителите е разпространението на обяви на хартия из сградата на университета. Този подход е доста примитивен в днешен ден. Гъвкавостта на програмата на Софийски университет и в частност Факултет по Математика и информатика позволява на студентите на винаги да присъстват задължително на лекции. По този начин обявите, разпространени в сградата на университета стават абсолютно безсмислени.

Най-достоверния източник на информация относно живота в университета за студентите е Факултетен студентски съвет. Работата на ФСС е да разпространява информация свързана именно със стипендии, обяви за работа, нужда да бъде подпомогнат преподавател или студент, благотворителност в университета и т .н. Решението на проблемът с достоверността и точността на информация е работещо. Проблемът в този подход обаче е къде да търсим обявите на ФСС, как да се свържем с тях за въпроси.

Секцията за новини на ФСС се намира на личният им сайт. Личният им сайт все още е в процес на разработка, затова членовете на ФСС информират студентите с обяви през социалните мрежи - Facebook, Instagram и т.н.

В момента каналите за информация са преплетени в порочен кръг, който задължава студентите и преподавателите да имат акаунт в някоя или всички социални мрежи, да проверяват всяко едно място, в което може да има информация и накрая да преглеждат официалните сайтове на организациите за още нужна информация.

Тук отново се натъкваме на проблема, ако ученик или преподавател знае за дадена стипендия или специалист по набиране на хора иска да публикува обява за работа сред студентите на университета, как да разпространи информацията, за да бъде видима в сайта на ФСС (когато бъде завършен) или да бъде публикувана от ФСС член - единственият начин за момента е да се свържем с ФСС чрез, след което да му предадем цялата информация.

## Критерий за сравнителен анализ

За да проектираме и създадем решение, което поправя недостатъците на тези, които текущо се използват трябва да изтъкнем предимствата и недостатъците на разгледаните решения.

След прегледа на предметната област и проблемите пред нея можем да дефинираме следните основни точки, чрез които да изградим критерий за сравнение.

Потребителски интерфейс

Механизмът за споделяне на информация трябва да предоставя удобен за ползване потребителски интерфейс, който да е достъпен през Интернет. Всеки студент, преподавател, специалист по набиране на служители трябва бързо и лесно да може да създава обяви и да ги споделя с останалите заинтересовани лица

Достъп

Достъпът до средствата за споделяне на информация трябва да лесно достижим от всеки потребител. Не трябва достъпът до средствата за споделяне на информация да е зависим от географските ширини, от които се опитваме да споделим обява.

Достоверност

Информацията за обявите, публикувани от различните настоящи средства трябва да е точна и достоверна. Да сочи към текущо активна и валидна обява. Съдържанието трябва да бъде описателно, сравнително кратко и ясно, да съдържа линк към официалния сайт на организацията или обявата. След като обява бъде публикувана, оторизирано лице може да провери за достоверността и точността на информацията и да предприеме действия по премахване ако е нужно.

Поддръжка на валидността и съдържанието на обявите

Съдържанието и периодът на валидност на разпространените обяви трябва да да могат да се редактират лесно и с няколко действия от страна на създаделя.

Независимост при публикуване на обява

Заинтересованите лица, които желаят да публикуват обяви чрез някое от предоставените средства не трябва да са задължени да имат взаимодействие с оторизираните лица, отговорни за разпространение на обяви в университета. Не трябва потребителите, които желаят да споделят информация да са зависими по какъвто и да е начин от членове на ФСС, служители в кариерен център на университета или администратори на официален сайт на университета.

Разширяемост

Броят на обявите, които се разпространяват в университета може да варира от 1 до хиляди. Обявите може да са от 1 или 2 категории, а могат да бъдат и в повече от 5 категории. Поради тази причина е изключително важно средството за споделяне на информация да може да поддържа голям обем от данни, които лесно и бързо да обработва. Публикуването на голям брой обяви трябва да бъде бърз и независим процес.

Осигуряване на индивидуална информация

Всяко заинтересовано лице трябва да може да получи информацията, която е важна за него по бърз и лесен начин. Не трябва потребителите да са принудени да откриват важни за тях обяви използвайки множество източници на информация и препратки, както и да им се налага да преценят коя обява е валидна за тях и коя не.

В тази точка са сравнени разгледаните решения в точка 2.3 спрямо критериите, описани в точка 2.4

Таблица 1: Сравнителен анализ на съществуващи решения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий / Решение | Официален сайт - секция “Кариерен център” | Официален сайт на ФСС | Социални мрежи - Facebook групи | Хартиени обяви |
| Потребителски интерфейс | Да - само за разглеждане | Да - само за разглеждане | Да | Не |
| Достъп | Да - само за разглеждане | Да - само за разглеждане | Да | Не - изисква се физическо присъствие |
| Достоверност | Да - публикуват се от админ | Да - публикуват се от член на ФСС | Не - няма средства за проверка на достоверност |  |
| Поддръжка | Да | Да | Да | Не |
| Независимост при публикуване | Не - потребител трябва да сподели информация с админ, за да е публична | Не - потребител трябва да сподели информация с ФСС, за да е публична | Да | Не - потребител трябва да сподели информация с административен служител |
| Разширяемост | Не - администратор трябва да създава всички обяви | Не - ФСС трябва да създава всички обяви | Да | Не - административен служител трябва да създава всички обяви |
| Индивидуализация на обяви | Не - обявите не са категоризирани | Не - обявите не са категоризирани | Не - обявите не са категоризирани | Не - обявите не са категоризирани |

## Изводи

В настоящата глава беше разгледана предметната област, бяха идентифицирани проблеми, които са характерни за нея и бяха очертани съществуващи методи, средства и системи за разрешаването на тези проблеми. Очертахме основни точки, които трябва да се следят при избора на средство или система за решаване на тези проблеми и оформихме критерий, по който да бъде направен сравнителен анализ за ефективността на настоящите методи и решения.

Като извод може да заключим, че нито едно от настоящите решения на отговаря на всички критеритерият за успешно решение. На абсолютно всеки настоящ подход му липсва основно изискване, а именно обяви да могат да се създават от всеки, с което базата от данни да бъде все по-пълна.

# Глава 3

В тази глава ще представим различните технологии, които могат да бъдат използвани за решаване на проблемите очертани във втора глава. Чрез сравнителен анализ ще изберем най-подходящите, ще представим техните предимства и недостатъци.

## Изисквания към средствата

Системата, която ще се разработи в настоящия труд трябва да представлява уеб-базирано приложение. Тя трябва да се състои от сървърна част, която извършва основните взаимодействия с базата данни, клиентска част, която има потребителски интерфейс, през който потребителят ще може да взаимодейства със системата.

Сървърната част ще е достъпна за клиента през интерфейс - API. API ще позволява на клиента да комуникира с базата от данни, да достъпва функционалности като добавяне на данни в базата от данни, четене на данни, обновяване на данни и изтриване на данни, без да излага на показ имплементацията си.

Уеб интерфейсът ще се достъпва от клиента използвайки HTTP протоколът и url-и за всяка една функционалност.

Към сървърната част е свързана и база от данни, която ще съхранява информация, както за обявите, така и за профилите на регистрираните потребители.

Потребителска клиентска част ще дава възможност на потребителите да могат да създават обяви от всеки компютър, притежаващ уеб браузър.

Сървърен език за програмиране

За сървърната част е необходимо да се изберат езици за програмиране от високо ниво, защото тя ще представлява REST API, което ще оперира с базата от данни. Изборът на език за програмиране трябва да е съобразен изцяло с технологията и рамките, която ще използваме.

Технологията и езикът за програмиране трябва да отговарят на следните изисквания:

* CRUD REST операции, за опериране с базата от данни
* Трябва да поддържа JSON обекти като формат за отговори и размяна на съобщения
* Трябва да поддържа ORM, за автоматично създаване на модели от базата данни.
* Да съществуват библиотеки за свързване с базата данни
* Да съществуват библиотеки за авторизация и автентикация на потребителите
* Да могат лесно да се управляват зависимости между пакети и библиотеки
* Стартирането на нов проект да става бързо и лесно
* Да са безплатни и open source

База данни

Цялата информация за потребители, потребителски профили, обяви, категории ще се съхранява в базата от данни. Основната функционалност на разработваната система ще бъде да визуализира обяви, да се създават обяви и да се филтрира нужната информация. На база основните функционалности на системата стигаме до извод, че основен критерий за избор на База от данни ще бъде висока производителност на писане и четене.

Базата данни трябва да отговаря на следните изисквания:

* Да са безплатни и open source
* Тя трябва да предоставя сигурност и поддръжка на данните при изпълнение на транзакции
* Гарантира работа 24/7
* Предоставя функционалности за поддържане на цялостност на данните, атомарност,консистентност, изолираност, трайна поддръжка на транзакции, поддръжка на транзакции с много версии, неограничено заключване на ниво ред
* Да не изисква много място в хард диска, както и RAM. Целта на дипломната работа е да се създаде прототип

Потребителска клиентска част

Клиентска част ще има потребителски интерфейс, през който потребителят ще може да взаимодейства със системата. Тя ще дава достъп на потребителите до функционалността на системата. С цел да бъде удобна за ползване и достъпна от всеки студент, преподавател или всеки, който се интересува от животът в университета, потребителската клиентска част трябва да отговаря на следните условия:

* Да дефинира потребителски графичен интерфейс
* Да предоставя лесен за употреба графичен интерфейс за създаване и разглеждане на обяви, за регистриране на потребители, за вход и изход от потребителски профил.
* Клиентът трябва да се състои от 2 основни панела: потребителски и администраторски.

Технологията и езикът за програмиране трябва да отговарят на следните изисквания:

* Да дават възможност за създаването на отделни компоненти, които предоставят достъп до функционалностите на системата.
* Да могат лесно да се проследяват и прихващат грешки при имплементацията още в браузъра или в средата за разработка (да могат да се дебъгват)
* Да се поддържат технологии и езици за създаване на графичен интерфейс
* Да поддържа MVC архитектура
* Да може да общува със сървърната част асинхронно, използвайки REST методология
* Да има възможност да се вземат асинхронно REST данни като се използва HTTP услугата. (subscribe)
* Да могат лесно да се управляват зависимости между пакети и библиотеки
* Стартирането на нов проект да става бързо и лесно
* Да са безплатни и с отворен код
* Да имат ясно дефинирана структура на файловете и проектните ресурси

Среда за разработка

Базирайки се на избора на езици и технологии за реализирането на сървърната и клиентската част, трябва да се изберат конкретни за тях среди за разработка на код. Избраната среда трябва задължително да поддържа програмния език, за който ще се използва, и да улеснява максимално неговото писане. Средата за разработка трябва да може да предоставя автоматични предложения за допълване на код, за генериране на код, за откриване на синтактични грешки и други полезни инструменти. Средата за разработка трябва да позволява лесно проследяване (дебъгване) на имплементацията, намиране на грешки и прихващане. Средата за разработка трябва да е безплатна и с отворен код.

Система за управление на кода

За да се проследи имплементацията на системата трябва да се използва система за управление на кода. Тя трябва да може да прихваща всяка промяна в различните видове файлове спрямо предишно състояние на кода. Трябва да може да пази информацията за всяка една итеративна промяна, да подсигури съществуването на различни версии на кода едновременно, лесното възвръщане на предишни версии, както и сливането и разделянето на тези версии. Системата трябва да поддържа преглед на историята на промените, както и осъществяване на промени от няколко разработчика с подходяща поддръжка за разрешаване на конфликтите помежду им. Системата за управление на кода трябва да е безплатна и с отворен код.Системата трябва да поддържа работа офлайн и да има бърза скорост на откриване на промени и извършване на сравнение.

Система за съхранение на кода

Кодът на системата трябва да бъде съхраняван централно и не само локално. Изискванията към системата за съхранение на кода са:

* Системата за съхранение на кода трябва да бъде налична по Интернет.
* Трябва да поддържа работа с избраната система за управление на кода.
* Трябва да бъде сигурна и да дава достъп до кода само на аутентикирани потребители.
* Трябва да е безплатна и с отворен код.

## Видове средства, начин и място за използването им – сравнителен анализ

Сървърен език за програмиране

Сървърната част на представлява основните функционалности, които ще изпълнява системата. Сървърната част ще извършва основни операции с базата от данни като четене, писане на информация, обновяване на информация, изтриване на информация от базата. Поради тази причина трябва да бъде избран език за програмиране, който поддържа средства, библиотеки и технологии за лесна, бърза и сигурна връзка с базата от данни, както и вградени функции за извършване на операциите.

Като най-известни и използвани езици за програмиране в бизнес средите и създаване на уеб системи и публични интерфейси (API) ще разгледаме Java и C#.

Първите версии на Java и C# имат много прилики. Тяхната основна идея е свързана с преминаването от езици от ниско ниво като C++ към езици от по - високо ниво, които се компилират до байтов код и могат да бъдат изпълнени на виртуална машина.

“**Java** е език за разработка на сървърни приложения, сериозна бизнес логика, големи индустриални системи, включващи сложна обработка и изискващи голяма надеждност. На Java се пишат както сложни server-side системи, така и **уеб базиран софтуер**. “ [4]

“C# е доста универсален език, с който се програмира всичко, но има сериозно ограничение – той **не е отворена платформа с отворен код**. Като собственост на Microsoft, езикът **C# се ползва предимно в Microsoft екосистемата**, т.е. **за програмиране под Windows** (сървърни приложения по Windows, cloud приложения в Windows среда, уеб приложения под Windows с ASP.NET и мобилни приложения в Windows Phone и Windows 8). Ако C# и .NET бяха отворени език и платформа (като Java и PHP), вероятно C# щеше да е най-популярният език за програмиране в света.” [4]

Java е популярна с поддържането на платформа за създаване на микросървиси и публични интерфейси - Spring Boot.

Spring Boot осигурява добра платформа за Java програмистите за разработване на самостоятелни и сложни spring приложения, които могат лесно да се стартират и изпълняват. Най - голямото предимство на Spring Boot е минималната нужда от конфигурации. Други предимства на Spring Boot за разработчиците са:

* лесно разбиране и разработване на Spring приложения
* увеличаване на продуктивността
* намаляване на времето за разработка
* той осигурява мощна пакетна обработка и управлява REST крайни точки
* улеснено е управлението на зависимостите
* дава гъвкави решения за конфигуриране на база от данни и транзакции върху базата от данни

При стартиране на нов проект с Spring Boot, той автоматично конфигурира новия проект, базиран на зависимостите, които сме добавили в единствен конфигурационен файл.

В границите на настоящата дипломна работа основно изискване към езика, който ще се използва са наличието на рамки и библиотеки, които предлагат за връзка с базата от данни и извършване на операции по четене, писане, обновяване и изтриване на информация.

В тази връзка Java и специално Spring Boot предлагат Hibernate програмна техника. Hibernate е рамка с отворен код, която предоставя услуги като изпъняване на заявки и поддръжка на данните и релациите в базата от данни. Hibernate съпоставя Java класове с таблици от базата данни, както и Java типове до типове в SQL. По този начин ORM-ът върши 95% от работата на програмистите, свързана с поддържане на данните.

Едно от предимствата на Hibernate е, че подсигурява опростен API, улеснен и разбираем достъп до повечето му функционалности.

За да използваме Hibernate не се нуждаем от някакви допълнителни настройки, достатъчно е при създаване на нов Spring Boot проект да добавим зависимости **hibernate.dialect, hibernate.connection.username, hibernate.connection.password**

Базите от данни, които поддържа Hibernate са:

* HSQL Database Engine
* DB2/NT
* MySQL
* PostgreSQL
* FrontBase
* Oracle
* Microsoft SQL Server Database
* Sybase SQL Server
* Informix Dynamic Server

C# е известен със своя Entity Framework като ORM. Entity Framework и всички последователни негови версии, както и Entity Framework Core са продукти на Microsoft. Първоначалните версии на Entity Framework поддържат само model-first и database-first методи за създаване на модели, съответстващи на обектите от базата от данни. Следващите версии на Entity Framework започват да поддържат и code-first подход и предлагат по - изчистена публичен интерфейс за използване. (API) Появата на .NET Core като продължение на версиите на EF носи нови функционалности на потребители на ORM средствата на Microsoft.

.NET Core 1 се има за идея цялостно да пренапише функционалностите на съществуващите решения. Затова първата версия реализира само някои основни функционалности плюс архитектура, която поддържа нерелационни бази от данни.

.NET Core 1.1 носи много важната функционалност устойчивост на връзката с базата от данни.

.NET Core 2 реализира функционалности като глобални филтри, операторът LIKE, DbContext pooling и т.н

Entity Framework се състои от множество DLL файлове, които идват под формата на NuGet пакети. Съществува зависимост между NuGet пакетите, което понякога се оказва огромен минус при използването, заради нуждата да се инсталират всички пакети, зависимости на един, който желаем да използваме.

Базите от данни, които поддържа Entity Framework са:

* SQL Server (including Azure)
* SQLite
* In-memory

Друга функционалност, която ще е нужна да имплементираме с програмните езици е регистриране на потребители, автентикация и авторизация.

Един от най-сигурните начини за автентикация и авторизация, както и лесен з аимплементация е JWT. JSON Web Token (JWT) е отворен стандарт, който дефинира компактен и самостоятелен начин за сигурно пренасяне на данни между отделните слоеве под формата на JSON обект. Тази информация може да бъде верифицирана, защото е електронно подписана. JWTs може да се подпише като се използва **HMAC** алгоритъм, публичен/частен ключ, използвайки **RSA** or **ECDSA**.

И двата програмни езика имат еднаква програмна силна при имплементацията на JWT механизмът. JSON Web Token-ът се състои от следните три секции:

* Header: JSON формат, който е кодиран в Base64
* Claims: JSON формат, който е кодиран в Base64
* Signature: Създаден и подписан базирайки се на предишните две секции, които са кодирани в в Base64

JWT се предава по HTTPs ако съхраняваме някакви чувствителни данни в Claims секцията. JWT може да съдържа различна информация: потребителско име, роли, време за изтичане валидността и т.н

База данни

Преди да бъдат разгледани конкретни опции за избор на база от данни, трябва да бъде анализирано какъв вид база данни би била най-подходяща за съхраняване на информацията относно потребители, обяви и т.н Базите от данни биват основно два типа - релационни и нерелационни. Нерелационните бази от данни имат по - проста схема т.к те съхраняват информацията в JSON формат и нямат ясна структура. За разлика от тях релационните бази от данни съхраняват информацията под формата на таблици, редове и колони, свързани с логическа връзка помежду си. Те имат ясно дефинирана структура.

“SQL базите се използват масово - при малки обеми информация например за уеб сайт с две страници, до големи уеб или мобилни приложения, блогове, онлайн магазини и други. Най-известните готови системи за управление на съдържанието (CMS) поддържат и използват релационни бази данни - WordPress, Joomla, Drupal, Magento и други. По-малко са тези обаче, които поддържат NoSQL бази данни (като Drupal).”[7]

Друго предимство на релационните бази от данни е, че поради добре дефинираната си структура те позволяват използването на сложни заявки. Операциите са атомарни, което означава че при възникване на грешка по време на обновяване, информацията в базата от данни ще остане в първоначалния си вид.

Базата от данни на настоящия прототип е предназначена за съхранение на информация, касаеща само СУ”Св. Климент Охридски”. Поради тази причина не се очаква тя да съхранява големи обеми от данни.

Известни релационни бази от данни са MySQL, MS SQL, Oracle и т.н.

Едно от основните изисквания на базата данни за настоящия труд е тя да бъде поддържана от програмният език, чрез който ще имплементираме WEB API. Трябва да се изследва и анализира производителната способност за четене и писане на базата, защото това са основните функционалности на системата за споделяне на информация.

Oracle e СУБД, предназначено за големи корпорации и голям обем от данни със сложни връзки между тях.

Базирайки се на тези знания ще бъде направен сравнителен анализ само между MySql и MS SQL за избор на подходяща система з ауправление на бази от данни.

MySQL е система за управление на бази от данни с отворен код, докато SQL Server е продукт на Microsoft. Microsoft дава възможност на потребителите си да избират между различни версии на SQL Server спрямо нуждите и бюджета си. И двете системи поддържат споменатите програмни езици за реализация на сървърната част на уеб системата.

MS SQL има само един механизъм за съхранение, докато MySQL има няколко. Наличието на няколко механизма за съхранение, които поддържа MySQL за разлика от единствения на MS SQL прави MySQL по-гъвкава и предпочитана.

Потребителите могат да избират от няколко версии на MS SQL Server според броя на едновременните потребители и изискванията на проекта. Те дори могат да интегрират СУБД с различни патентовани и отворени технологии. Но MySQL е проектиран с функции, допълващи нуждите на съвременните уеб приложения. Много разработчици на уеб приложения използват MySQL, заради лесната му съвместимост с операционната система Linux и уеб сървъра Apache.

Потребителска клиентска част

Изискване към разработваната система е тя да бъде достъпна през Интернет, използвайки уеб браузър.

За да се разработи потребителски интерфейс, достъпен през уеб браузър обикновено се използват езици за програмиране като JavaScript, HTML5,Cascading Style Sheets (CSS). Потребителската клиентска част обикновено обединява тези програмни езици. Заради голямото разнообразие от рамки и библиотеки базирани на JavaScript е много важно да се направи анализ на някои от тях и да се избере най-подходящата за текущия проект.

Като най-популярни рамки, базирани на JavaScript ще разгледаме Angular и React.

Angular е рамка с отворен код, разработена от Google за създаване на динамични, модерни уеб приложения. Печели много поддръжници с това че успява да премахне ненужния код и осигури по-леки и по-бързи приложения. Angular днес се използва от повече от 44% от софтуерните компании.

Тази рамка помага да се създават интерактивни и динамични сайтове, предоставяйки средства за шаблонизиране, двупосочно свързване, разделяне кода на модули, изпращане на заявки към RESTful API, dependency injection и възможност за създаване на AJAX заявки.

Angular приложенията за имплементирани като се използва програмния език TypeScript. Това е скриптов език, базиран на JavaScript, който подсигурява висока сигурност и поддържа типове (примитивни, интерфейси и т.н). Подпомага прихващането и отстраняването на грешки.

За разлика от други езици като CoffeeScript или Dart, TypeScript не е самостоятелен език. Предимство на TypeScript е, че може директно да се дебъгва в браузър или който и да било редактор. Angular използва HTML, за дефиниране на потребителския интерфейс. HTML, в сравнение с JavaScript е по - опростен и дален не е толкова заплетен. Той е декларативен и интуитивен език. Рамката е подобрена и с добавянето на шаблонът MVC (Model-View-Controller). Angular организира кода по такъв начин, че да е спазен шаблонът. Кодът е организиран в компоненти, модули, услуги или директиви. Това позволява преизползваемост на кода, лесно разделение, лесна поддръжка и разпределение между членове на екип.

React познат още като React.js или ReactJS е библиотека на JavaScript за създаване на потребителски интерфейс. Поддържа се от Facebook и група от индивидуални разработчици и компании. Той се използва при разработката на сайтове или мобилни приложение. Поддържа рендериране на данни. По този начин изработването на React приложения обикновено изисква използването на допълнителни библиотеки за управление на състоянията, навигация, взаимодействие с API и т.н.

Мнението на програмистите е, че React не е толкова интуитивен и отнема повече време за усвояване.

Ако използваме React JS с HTML, страниците ни ще се обновяват, използвайки DOM. Това работи добре при прости, статични страници, ноз а динамични уебсайтове, с взаимодействие от потребителя може да бъде проблем, т.к DOM трябва да презарежда всеки път при всяко събитие.Кодът при React също е разделен на компоненти.

Разликата между JavaScript библиотека като React и JavaScript рамка като Angular е че, когато имаме библиотекаа, разработчиците прилагат библиотечен код в отделни инстанции, които се извикват. Когато става въпрос за рамки обаче, рамката създава скеле, което подрежда уебсайта или приложението и предоставя специални области за код, които да бъдат имплементирани.

React JS e проект с отворен код, което означава, че всеки може да изтегли кода му и да го модифицира. Това също означава, че какъвто и от специфичен компонент или функция на потребителския интерфейс да имаме нужда, има възможност да я открием в някоя React библиотека. По този начин броя на библиотеките може да нарасне неимоверно. Точно затова React не се препоръчва за системи с много страници.

Предимството на Angular e, че няма нужда от много допълнителни библиотеки, т.к е пълноценна рамка. Структурата на React осигурява на разработчиците свободата да избират. Няма единствена правилна структура за React приложение. Това се оказва трудност в началото или дизайна на едно приложение. Разнообразната структура също и проблем при читаемостта на кода или въвеждане на нов разработчик в нов проект. React предлага само View layer от (MVC), а моделът и контролерът се добавят като се използват допълнителни библиотеки.

От друга страна структурата на Angular e фиксирана. Базирана е на MVC шаблонът.

Кодът на едно приложение е разделен в компоненти. Всеки компонент се състои от 4 файла:

* файл с TypeScript код, който имплементира компонента
* файл с HTML скрипт, който дефинира изгледът на потребителския интерфейс
* файл с CSS код - който съдържа стилизацията на изгледа
* специален файл, подпомагащ тестването

Когато има предварително заложена структура, навлизането в нов проект е лесно, поддръжката на код е лесна, започването на нов проект е бързо и лесно.

Синхронизацията на данни между моделът и изгледът в React е еднопосочна и се осъществява чрез Redux библиотеката.

Синхронизацията на данни между моделът и изгледът в Angular е двупосочна , което улеснява процесът на разработка.

Среда за разработка

Средата за разработка зависи от избрания език за програмиране. Различните среди имат различна поддръжка на съответен език.

Т.к сървърната част ще бъде имплементирана само на Java или C#, най - популярните среди за тези езици са:

* За Java средите с най - добра поддръжка са Eclipse и IntelliJ Idea. Eclipse e изцяло безплатен, докато за IntelliJ трябва да се закупи лиценз. IntelliJ предлага и безплатна версия, но някои функционалности са липсващи и ограничени. И двете среди имат почти еднакви функционалности за поддръжка на Java. Основно нещо, което ги различава са системните изисквания:

IntelliJ IDEA може да се инсталира на Windows, MacOS и Linux със следните хардуерни изисквания:

* 2 GB RAM минимум, 4 GB RAM препоръчително
* 1.5 GB място в хард диск + поне 1 GB за кеш
* 1024×768 минимална резолюция на екрана

Eclipse IDE може да се инсталира на всичко с JVM включвайки Windows, MacOS, Linux и Solaris. Хардуерните изисквания са:

* 0.5 GB RAM минимум, 1+ GB RAM препоръчително
* 300 MB място в хард диск , 1+ GB препоръчително
* За C# средата с най-добра поддръжка е Visual Studio - “Visual Studio предоставя мощна интегрирана среда за писане на код, компилиране, изпълнение, дебъгване (както за високо така и за машинно ниво), тестване на приложения, дизайн на потребителски интерфейс (форми, диалози, уеб страници, визуални контроли и други), моделиране на данни, моделиране на класове, изпълнение на тестове, пакетиране на приложения и стотици други функции. Могат да се добавят и плъгини, които повишават функционалността на почти всяко ниво – включително добавянето на поддръжка заsource-control системи. Microsoft предлага безплатни „Express“ издания на някои от компонентите на Visual Studio“ [9]

Хардуерните изисквания на средата са:

* 1.8 GHz бързина на процесора.
* 2 GB RAM; 8 GB RAM препоръчително
* Минимум 800MB дисково пространство.

Среди за разработка на SQL скриптове има безброй много. Най - препоръчително е обаче средата да се избере спрямо избраната база от данни

* За MySQL една от най - популярните среди е MySQL Workbench. Тя се инсталира с инсталирането на MySQL сървърът. Поддържа лесен и интуитивен потребителски интерфейс за разглеждане елементи на базата, разглеждане на данни, създаване на таблици, тригери, индекси и т.н Предлага и редактор за писане на SQL скриптове.
* За MS SQL най - използваната среда е SQL Server Management Studio.То се използва за конфигуриране, управление и администриране на всички компоненти в SQL Server. Съчетава широка група от графични инструменти с голям брой текстови редактори, осигуряващи на разработчиците и администраторите всички нива на достъп до сървъра.

Потребителската Клиентска част ще бъде имплементирана с езиците JavaScript, HTML и CSS.

* За JavaScript/HTML/CSS популярни среди са Eclipse (с плъгин CоdeMix) Atom и Visual Studio Code. Visual Studio Code и Atom са абсолютно безплатни, докато Eclipse и по - точно CоdeMix има безплатен тестов период, след което трябва да се закупи лиценз. И трите среди имат достатъчно добра поддръжка на езиците. Неудобството при Visual Studio Code е, че за да бъде настроен един проект за дебъг се изискват допълнителни конфигурационни файлове (launch.json), докато Eclipse сам създава тези конфигурационни файлове.

Система за управление на кода

Най - използваните системи за управление на кода са SVN и Git.

SVN е безплатна и с отворен код система. Разпространява се с лиценз на Apache. SVN има по-бавно скорост на сравнение и липса на разпределен контрол на ревизията. Разпределеният контрол на ревизията използва peer-to-peer модел, вместо да използва централизиран сървър за съхранение на обновения код. SVN има няколко недостатъка, които го правят по - малко предпочитаната система за управление на кода:

* SVN все още има бъгове, свързани с преименуването на файлове и директории.
* Недостатъчно команди за управление на репозиторитата
* Малка скорост на сравнение

Оригиналната идея на Git е да бъде по-бърз от съществуващите системи.

Разпределените системи за контрол на ревизията, опровергават изцяло практиките, използвани в CVS. Първоначално Git е създаден за Linux и поради тази причина има най-високо бързодействие на тази ОС. В допълнение върви и на системи подобни на Unix и Windows. Git идва оборудван с широк спектър от инструменти, които помагат на потребителите лесно да се ориентират в системата, както и лесна навигация в историята на системата. Всяка инстанция на източника съдържа цялото дърво на историята, което може да бъде полезно, когато разработваме без интернет връзка.

Git се характеризира с:

* огромно подобрение на обработващата скорост
* лесни операции в бранча
* налична пълна история на дървото, дори и офлайн
* разпределен, peer-to-peer модел

Основната разлика между двете системи - SVN поддържа само едно хранилище на код, което е отдалечено и централно, докато Git поддържа и локални хранилища, което значително улеснява офлайн работата.

Процесът по създаване на различни branch-ове в SVN е далеч по-сложен от процеса в Git.

Система за съхранение на кода

Система за съхранение на кода за SVN няма. Централното съхранение на кода става на машини, притежавани от разработчиците. Този подход е изключително несигурен и примитивен.

За Git има популярна централизирана система за съхранение на код – GitHub.

GitHub е хранилище за съвместен код, който съхранява и преглежда кода, управлява проекти и създава изпълним файл на даден софтуер. Предлага всички функции на разпределени системи за контрол на версиите, функционалностите на системи за управление на кода, както и добавя собствени функционалности. Предоставя още функционалности като следене за бъгове, управление на задачите, както и информация за всеки проект.

Функционалността GitHub Actions дава възможност части от кода да могат да бъдат изпълнявани в контейнер, използвайки широката гама обръщения към GitHub API. Това дава възможност на потребителите, да оркестрират техния работен поток, базиран на всяко възникнало събитие. GitHub предлага CI/CD, което прави по-лесна автоматизацията на процесите по създаване на изпълним файл, тест, разгръщане на проектите. Работните потоци биват изпълнявани на виртуална машина.

GitHub не предлага платформа за разгръщане и се нуждае от допълнителни приложения като Heroku.

GitHub предлага безплатна версия, но с ограничени възможности. Тя основно се използва за приложения с отворен код.

## Избор на средствата

Средства за сървърната част - сравнителен анализ

|  |  |  |
| --- | --- | --- |
| Изискване / език за програмиране | Java | C# |
| Да могат да се реализират CRUD REST операции | Да | Да |
| Да поддържа JSON обекти | Да | Да |
| Да поддържа ORM | Да - Hibernate | Да - EF |
| Да съществуват библиотеки за свързване с базата данни | Да | Да |
| Да съществуват библиотеки за авторизация и автентикация на потребителите | Да | Да |
| Да могат лесно да се управляват зависимости между пакети и библиотеки | Да | Не - зависимостите между NuGet пакетите |
| Стартирането на нов проект да става бързо и лесно | Да | Не |
| Да са безплатни и open source | Да | Не |

C# не отговаря на някои от основните изисквания за текущия труд. Трудното управление на зависимостите може да създаде огромни проблеми при реализация на проекта. Множеството версии и изисквания на ORM-ът Entity Framework също могат да спрат лесната имплементация на системата за споделяне на университетска информация.

База от данни - сравнителен анализ

|  |  |  |
| --- | --- | --- |
| Изискване / база от данни | MySQL | MS SQL |
| Да са безплатни и open source | Да | Не |
| Да предоставя сигурност и поддръжка на данните при изпълнение на транзакции | Да | Да |
| Да гарантира работа 24/7 | Да | Да |
| Да предоставя функционалности за поддържане ACID на данните | Да | Да |
|  |  |  |
| Добра интеграция с езика, избран за реализация на сървърната част | Да | Да |

На база сравнителния анализ избраната база от данни, която ще се използва е MySQL. Изборът е базиран на:

* Факта,че MySQL е безплатна и с отворен код, докато MS SQL като част от продуктите на Microsoft не в някои от версиите си.
* MySQL има много по-малки хардуерни изисквания, в сравнение с MS SQL

Потребителска клиентска част - сравнителен анализ

|  |  |  |
| --- | --- | --- |
| Изискване / технология | Angular | React |
| Да дават възможност за създаването на отделни компоненти | Да | Да |
| Лесно дебъгване и дебъгване в браузър | Да | Да |
| Да поддържа MVC архитектура | Да | Не |
| Да може да общува със сървърната част асинхронно, използвайки REST методология | Да | Да |
| Да могат лесно да се управляват зависимости между пакети и библиотеки | Да | Не |
| Стартирането на нов проект да става бързо и лесно | Да | Не |
| Да бъдат безплатни и с отворен код | Да | Да |
| Фиксирана структура на проекта | Да | Не |
| Да могат да се създадат множество страници, с лесна навигация между тях | Да | Не |

На база проведения сравнителен анализ изборът на Angular като рамка и TypeScript като език се оказва най-удачен.

Среда за разработка - сравнителен анализ

Т.к беше избран езикът за програмиране на сървърната част Java, то ще бъде проведен анализ между средите за разработка на код на Java, а именно Eclipse и IntelliJ IDEA

|  |  |  |
| --- | --- | --- |
| Изисквания / среда за разработка | Eclipse | IntelliJ IDEA |
| Трябва да поддържа програмния език, за който ще се използва | Да | Да |
| Трябва да може да предоставя автоматични предложения за допълване на код | Да | Да |
| Трябва да може да предоставя автоматични предложения генериране на код | Да | Да |
| Трябва да може да предоставя автоматични предложения за откриване на синтактични грешки | Да | Да |
| Трябва да позволява лесно проследяване (дебъгване) на имплементацията | Да | Да |
| Трябва да е безплатна и с отворен код | Да | Не |
| Да изисква възможно по-малко RAM и свободно място в хард диска | Да | Не |

Т.к беше избрана MySQL база от данни, то средата за разработка на SQL скриптове и разглеждане обектите на базата данни ще бъде MySQL Workbench, която се инсталира при инсталирането на сървъра.

Среда за разработка на потребителската клиентска част - сравнителен анализ. Подбрахме Angular и езикът TypeScript

|  |  |  |
| --- | --- | --- |
| Изисквания / среда за разработка | Eclipse | Visual Studio Code |
| Трябва да поддържа програмния език, за който ще се използва | Да | Да |
| Трябва да може да предоставя автоматични предложения за допълване на код | Да | Не - нужен плъгин |
| Трябва да може да предоставя автоматични предложения генериране на код | Да | Да |
| Трябва да може да предоставя автоматични предложения за откриване на синтактични грешки | Да | Да |
| Трябва да позволява лесно проследяване (дебъгване) на имплементацията | Да | Частично -допълнителни настройки |
| Трябва да е безплатна и с отворен код | Не - CodeMIX е платен | Не |
| Да изисква възможно по-малко RAM и свободно място в хард диска | Не | Да |

На база резултатите от проведения анализ и допълнителния факт, че Eclipse е избраната среда за разработка на сървърната част е логично да се избере отново Eclipse и за среда за разработка на Клиентската част. Това премахва нуждата от инсталиране, настройване и работа в две среди за разработка.

Система за управление на кода - сравнителен анализ

|  |  |  |
| --- | --- | --- |
| Изискване / система | SVN | Git |
| Да прихваща всяка промяна в различните видове файлове спрямо предишно състояние на кода | Да | Да |
| Да подсигури съществуването на различни версии на кода едновременно | Да | Да |
| Лесното възвръщане на предишни версии | Не | Да |
| Лесно сливането и разделянето на тези версии | Не | Да |
| Да поддържа преглед на историята на промените | Не | Да |
| Да поддържа осъществяване на промени от няколко разработчика с подходяща поддръжка за разрешаване на конфликтите помежду им | Не | Да |
| Да е безплатна и с отворен код | Да | Да |
| Да поддържа работа офлайн | Не | Да |

Базирайки се на проведения анализ Git е избраната система за управление на кода.

Система за съхранение на кода

Базирайки се на фактът, че SVN не предлага система за съхранение на код, а и избраната система за управление на код е Git, избираме системата за съхранение на код да бъде GitHub.

## Изводи

В началото на главата дефинирахме изискванията, на които трябва да отговарят средствата и технологиите, които ще използваме за разработка на системата. Предоставихме информация за най - популярните технологии за съответните части на системата. На база предоставените изисквания и доколко всяка технология удовлетворява тези изисквания направихме сравнителен анализ между подбрахме средства, с които ще бъде разработена системата.

# Глава 4

## Концептуален модел

Настоящата дипломна работа има за цел да реализира система, която позволява лесната и удобна обмяна на информация между заинтересованите лица в СУ “Св. Климент Охридски”. Потребителите, които ще използват системата ще могат да преглеждат и споделят информация под формата на обяви, които ще бъдат разпределени в три категории: обяви за работа, обяви за стипендии и обяви за благотворителност. Потребителите ще могат да създават собствени акаунти, спрямо своят статус в университетското общество: т.е те могат да бъдат специалисти от компании, които се интересуват от набиране на кадри от университета. преподаватели, студенти или членове на университетската администрация. Акаунтите ще бъдат попълвани спрямо интересът на потребителя, а именно дали е работник, който се интересува от обявите за работа, дали е студент, който се интересува от обявите за стипендии или е потребител, който се интересува от обявите за благотворителност. Администратор на системата ще се грижи за поддържане на информацията в системата.

# Библиография

1. <http://kmk.fmi-plovdiv.org/InternetTechnology.pdf>
2. <https://bg.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B5%D0%BD_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81#%D0%A3%D0%B5%D0%B1_API>
3. <https://www.fmi.uni-sofia.bg/bg/node/2071>
4. <https://nakov.com/blog/2013/06/30/koy-ezik-za-programirane-da-uchim-php-java-csharp-javascript-sql/>
5. <https://www.tutorialspoint.com/hibernate/hibernate_overview.htm>
6. <https://jwt.io/introduction/>
7. <https://help.superhosting.bg/sql-nosql-databases.html>
8. <https://www.grazitti.com/blog/8-proven-reasons-you-need-angular-for-your-next-development-project/>
9. <https://bg.wikipedia.org/wiki/Microsoft_Visual_Studio>
10. <https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/>
11. <https://about.gitlab.com/devops-tools/github-vs-gitlab.html>