UniVerse

CONNECTING MINDS, BUILDING FUTURES

Архитектурен проект на UniVerse

Автори:

Михаил Щерев

Силвио Стойков

Валентин Хаджиминов

Стефан Ангелов

Дата: 28.10.2023г.

Съдържание

1. Въведение

Университетската социална мрежа "UniVerse" е иновативна платформа, предназначена да обедини студентската и преподавателската общност в едно виртуално пространство. Целта е да се стимулира активна комуникация, сътрудничество и обмен на ресурси в рамките на университетския живот.

Докато функционалностите на приложението се определят от нуждите на потребителя, тяхното внедряване и поддръжка изискват подходяща архитектурна структура. Този архитектурен проект представя детайлното устройство на "UniVerse", обхващащо както клиентската (frontend), така и сървърната (backend) части на системата, базата данни, както и външните интеграции.

Архитектурният план е предназначен да служи като справочен документ за всички разработчици, архитекти и други заинтересовани страни, участващи в разработването, разширяването или поддръжката на UniVerse. Той предоставя пълна картина на структурната организация на системата, включително основните компоненти, техните взаимовръзки и начина им на взаимодействие.

В следващите раздели ще представим подробно всеки един от компонентите на системата, тяхната роля и начин на работа, както и ключовите архитектурни решения, реализирани в дизайна на "UniVerse".

1.1. Участници в проекта

1.1.1. Фронтенд разработчик

Основни задачи и отговорности:

- Дизайн и реализация на потребителски интерфейс: Превръщане на дизайнерски макети във функционални уеб страници.
- Интеграция с бакенд: Взаимодействие с API и други бакенд услуги.
- Оптимизация: Гарантиране на бързо зареждане и висока производителност на уеб страницата.
- Тестове: Провеждане на фронтенд тестове за браузърна съвместимост.
- Технологии и инструменти: JavaScript фреймуърк: **React**. **CSS** и предпроцесор: **SASS**. Инструменти: **Vite, npm** и др.

1.1.2. Бакенд разработчик

Основни задачи и отговорности:

- Създаване на API: Проектиране и разработване на API, чрез което фронтендът да комуникира с базата данни и други услуги.
- Логика на приложението: Писане на кода, който ще управлява функционалностите на приложението.
- Интеграция: Свързване на приложението с външни услуги и **Moodle** платформата.
- Осигуряване на сигурността: Предпазване на приложението от потенциални пробиви на сигурността и атаки.
- Технологии и инструменти:
    - Езици: **Java**, **Spring Boot, Hibernate**

1.1.3. Разработчик на база данни

Основни задачи и отговорности:

- Създаване и поддържане: Проектиране на структурата на базата данни и управляване на нейната ефективна работа.
- Оптимизация: Осигуряване на ефективни заявки към базата данни и нейната оптимална работа.
- Бекъп и възстановяване: Редовно съхраняване на резервни копия на данните и осигуряване на надеждни стратегии за възстановяване при необходимост.
- Сигурност: Проектиране и прилагане на политики и практики за защита на данните.
- Технологии и инструменти:
    - Системи за управление на бази данни: **PostgreSQL**.
    - Инструменти: **PGAdmin**

1.1.4. UI/UX дизайнер

Основни задачи и отговорности:

- Създаване на прототипи: Изработване на груби версии на интерфейса, които да помогнат в разработката и тестването на дизайна.
- Дизайн на интерфейс: Определяне на как приложението ще изглежда - избиране на цветове, шрифтове и други графични елементи.
- Изготвяне на потребителски пътеки: Проектират последователността на действията, които потребителите извършват в приложението, за да постигнат конкретна цел.
- Тестване на дизайна: Тестване на ефективността на дизайна и правене на необходими корекции.
- Технологии и инструменти:
    - Програми за дизайн: **Figma**.

1.1.5. Софтуерен архитект

Основни задачи и отговорности:

- Дефиниране на структурата: Определяне на ключовите компоненти на системата, техните взаимоотношения и как те взаимодействат помежду си.
- Техническа стратегия: Избиране на подходящите технологии, платформи и инструменти, които да отговарят на изискванията на проекта.
- Оптимизация: Гарантиране, че системата е проектирана по начин, който позволява лесно мащабиране, висока производителност и надеждност.
- Сигурност: Внедряване на най-добрите практики за сигурност на софтуера, защитаващи данните и функционалността на системата от потенциални угрози.
- Сътрудничество: Работа в тясно сътрудничество с други ключови роли, като разработчиците, UI/UX дизайнерите и продуктовите мениджъри, за да се гарантира, че техническата визия съвпада с потребителските и бизнес изисквания.

1.1.6. QA (Quality Assurance) специалист

Основни задачи и отговорности:

- Планиране на тестове: Определяне на тестовите случаи, базирани на изискванията и спецификациите на проекта.
- Ръчно тестване: Ръчно изпълнение на тестови сценарии за откриване на проблеми или несъответствия.
- Автоматизация на тестове: Създаване на автоматизирани тестове, които могат да бъдат изпълнени многократно без човешка намеса.
- Документация: Регистриране на откритите проблеми и предоставяне на подробно описание за тях, така че разработчиците да могат да ги коригират.
- Сътрудничество: Работа в тясно сътрудничество с разработчиците, за да се уверят, че всички проблеми са разрешени.

2. Предназначение на архитектурния проект

2.1. Обхват

2.1.1. Планиране

- Оценка на текущите нужди и изисквания на приложението.
- Изследване на текущите системи и платформи, като Moodle, за да се определи най-добрият начин на интеграция.
- Определение на ресурсите: колко разработчици, дизайнери и QA специалисти ще са необходими?
- Закрепване на времеви рамки за всяка следваща фаза на проекта.

2.1.2. Анализ на изискванията

- Проучване на потребителски сценарии и случаи на употреба, включително как студентите ще комуникират, как ще създават и споделят съдържание и т.н.
- Определение на основните функционалности: чат, създаване на постове, интеграция с Moodle за удостоверяване и др.

2.1.3. Дизайн

- Архитектурно планиране: Определяне на архитектура, база данни и технологии, които ще бъдат използвани.
- UI/UX дизайн: Създаване на wireframes и прототипи на интерфейса, определение на цветова палитра, стил и шрифтове.
- Планиране на инфраструктурата: Къде ще бъде хоствано приложението, какви са нуждите от мащабируемост и др.

2.1.4. Програмиране

- Фронтенд: Разработване на потребителския интерфейс, интеграция с бекенд услугите.
- Бекенд: Създаване на API-та, обработка на данни, интеграция с Moodle и други необходими системи.
- Бази данни: Оптимизация на структурата на данните, гарантиране на безопасността и интегритета на информацията.

2.1.5. Тестване

- Функционални тестове: Проверка дали всички функции работят според изискванията.
- Производителност и натоварване: Тестване на приложението при различни условия и обеми на трафика.
- Сигурност: Проверка за потенциални уязвимости и атаки.

2.1.6. Деплоймънт

- Избор на подходяща платформа за разпространение (напр. cloud услуги).
- Оптимизация на приложението за продуктивна среда.
- Мониторинг на приложението след пускането му в експлоатация.

2.2. Актьори

2.2.1. Студенти

- Feed система: Разглеждане на постове, новини и събития.
- Система за преглеждане на обяви за работа: Разглеждане и кандидатстване на обяви за работа.
- Профилна система: Управление на лични данни, настройки, постижения и курсове.
- Система за известия: Получаване на известия за предстоящи задачи и събития
- Система за обучение: Достъп до лекции, материали и оценки.
- Чат и комуникация: Взаимодействие с други студенти, преподаватели и групи.

2.2.2. Преподаватели

- Система за управление на курсове: Създаване на материали, управление на курсове.
- Чат и комуникация: Взаимодействие със студенти и други преподаватели.

2.2.3. Администратори

- Админ панел: Управление на потребители, контрол на съдържанието, постовете и мониторинг на системата.
- Система за интеграция: Управление на интеграции, като тази с Moodle, и други външни системи.

2.2.4. Работодатели

- Създаване на обява за работа: Публикуване на нова обява за работа.

2.2.5. Групови организатори

- Система за управление на групи: Създаване, модерация и управление на групови дискусии и материали.

2.2.6. Гости

- Публичен интерфейс: Достъп до публично съдържание, търсачка, информация за регистрация.

2.3. Използвани термини и символи

Термин	Значение
Entity	Обект, който се използва за моделиране и представяне на обекти от реалния свят или абстрактни концепции в софтуера.
HTTPS заявки	Протоколи, използвани за предаване на данни през интернет.
API	Набор от правила и протоколи, които позволяват на различни софтуерни системи да си взаимодействат и да споделят данни или функционалност.
Интерфейс	Структура, която определя набор от методи (функции) и техните сигнатури, но не и техните действителни реализации. Интерфейсите служат като план или шаблон за класове.

2.4. Източници

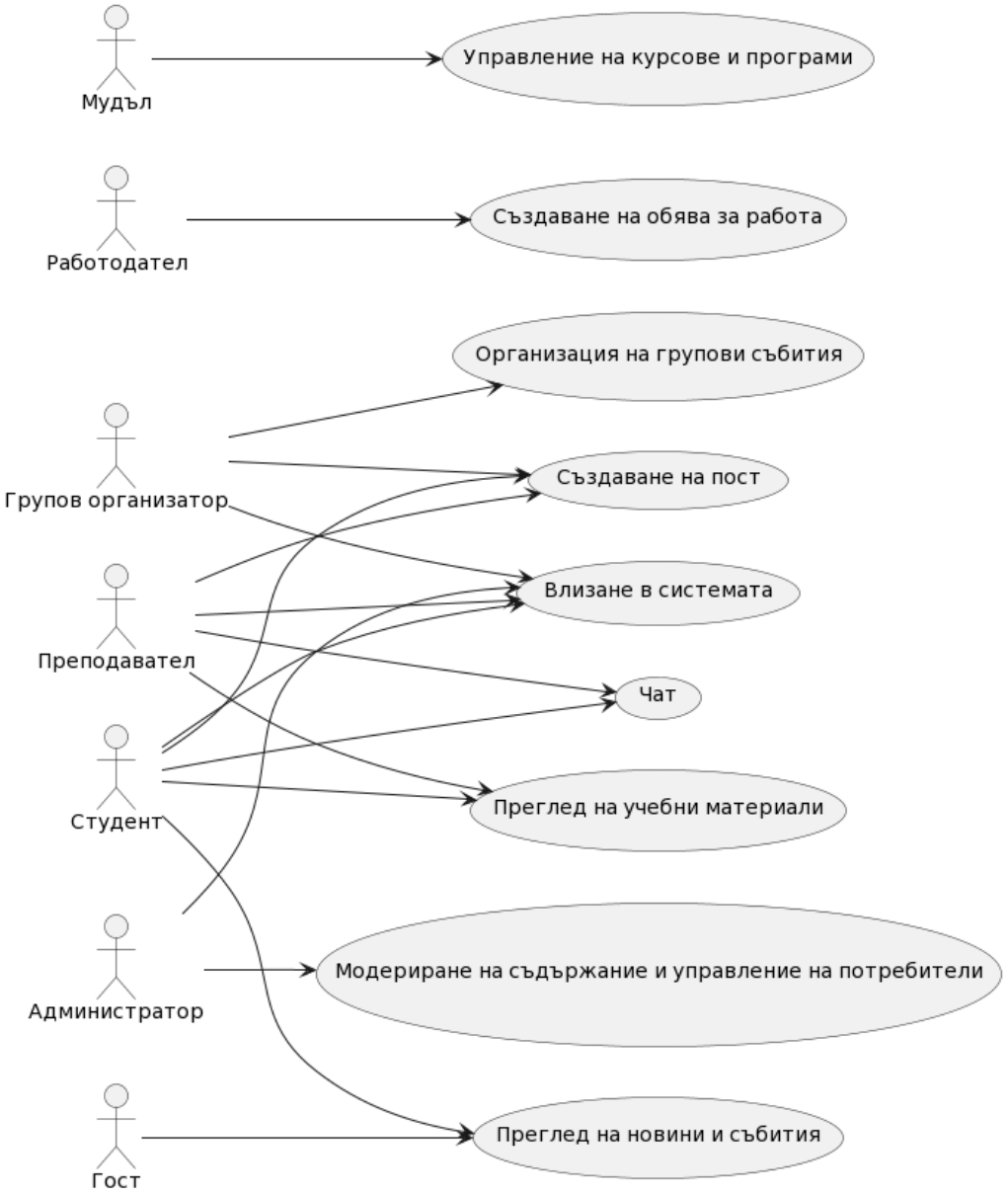
- https://www.couchbase.com/blog/application-development-life-cycle/
- https://www.ideamotive.co/blog/software-architecture-design-best-practices-you-should-know
- https://drawio-app.com/blog/uml-diagrams/
- https://plantuml.com/
- https://en.wikipedia.org/wiki/Extensibility (ревизия от 1 октомври 2023)
- https://nordic-backup.com/blog/how-to-achieve-high-availability-architecture/
- https://appmaster.io/blog/scalability-and-performance-software-architecture
- https://www.redhat.com/architect/nonfunctional-requirements-architecture
- https://appmaster.io/blog/security-in-software-architecture-design
- https://www.geeksforgeeks.org/software-testability/
- https://dev.to/sardarmudassaralikhan/interoperability-in-system-design-and-architecture-3n9

3. Архитектурен обзор

В настоящия документ ще бъде представена архитектурата на "UniVerse" чрез различни изгледи, които илюстрират как системата е организирана и функционира. Всеки изглед е насочен към конкретен аспект на системата и предоставя детайлна информация за съответните компоненти, отговорности и взаимодействия.

3.1. Use-case изглед

Представя основните сценарии на използване на системата от гледна точка на потребителите.



Роли на потребителите:

- **Студент**: Представлява университетски студент
- **Преподавател**: Представлява университетски преподавател
- **Администратор**: Администраторът е отговорен за управлението на системата и потребителите.
- **Групов организатор**: Този тип потребител има специфични функции за организация на групови събития и дейности.
- **Работодател**: Потребител, представляващ работодател, който може да публикува обяви за работа.
- **Гост**: Потребител, без регистрация, който може да вижда само новините и събитията.

Активности:

- **Влизане в системата**: Всички типове потребители трябва да влязат в системата чрез своите уникални идентификационни данни, за да използват платформата.
- **Създаване на пост**: Потребителите могат да създават публикации или постове в системата, като те могат да бъдат видими за други потребители.
- **Чат**: Потребителите имат възможност да комуникират чрез чат, като тази активност е налична за студенти, преподаватели и групови организатори.
- **Преглед на учебни материали**: Преподавателите и студентите могат да преглеждат учебни материали, свързани с различни курсове.
- **Управление на курсове и програми**: Курсовете и програмите ще бъдат предоставени от Мудъл.
- **Преглед на новини и събития**: Потребителите могат да преглеждат новини и събития, свързани с университета.
- **Модериране на съдържание и управление на потребители**: Администраторът има права за модериране на съдържание и управление на потребителски профили.
- **Организация на групови събития**: Груповите организатори могат да организират и управляват групови събития.
- **Създаване на обява за работа**: Работодателите могат да публикуват обяви за работа, които студентите могат да разглеждат.

3.2. Логически изглед

Логическият изглед на системата има за цел да предостави структуриран и детайлен поглед върху основните компоненти (класове, модули, пакети) и тяхнато взаимодействие. Този изглед се фокусира върху функционалните аспекти на системата, без да влиза в подробности относно конкретната имплементация или дистрибуция на компонентите.

Пакет "User Management"

User:

Атрибути:

- **userID**: Уникален идентификатор на потребителя.
- **username**: Потребителско име за достъп до системата.
- **password**: Парола за достъп.
- **email**: Електронна поща на потребителя.

Методи:

- **register()**: Регистрация на нов потребител.
- **login()**: Вход в системата с потребителско име и парола.

Guest (Гост):

Методи:

- **browsePosts()**: Преглед на публикации без нужда от регистрация.

RegisteredUser (Регистриран Потребител):

Методи:

- **createPost()**: Създаване на публикация.
- **startChat()**: Започване на чат с други потребители.

GroupOrganizer (Групов Организатор):

Методи:

- **createEvent()**: Организиране на групово събитие.

Employer (Работодател):

Методи:

- **publishJobOffer()**: Метод, чрез който работодателят може да публикува нова обява за работа.
- **viewApplicants()**: Метод, който позволява на работодателя да преглежда списък с потребители, които са кандидатствали за дадена позиция.

Пакет "Content Management" (Управление на Съдържанието):

Post (Публикация):

Атрибути:

- **postID**: Уникален идентификатор на публикацията.
- **title**: Заглавие на публикацията.
- **content**: Съдържание на публикацията.
- **date**: Дата на публикуване.

Методи:

- **create()**: Създаване на нова публикация.
- **delete()**: Изтриване на публикация.

Course (Курс):

Атрибути:

- **courseID**: Уникален идентификатор на курса.
- **name**: Име на курса.
- **description**: Описание на курса.
- **materials<>**: Списък с материали, свързани с курса.

Material (Материал):

Атрибути:

- **materialID**: Уникален идентификатор на материала.
- **title**: Заглавие на материала.
- **content**: Съдържание на материала.

JobOffer (Обява за работа):

Атрибути:

- **offerID**: Уникален идентификатор на обявата.
- **title**: Заглавие или позиция на работната обява.
- **description**: Подробно описание на обявата, включващо задължения, отговорности и друга важна информация.

- **requirements**: Изисквания към кандидатите, като образование, опит, умения и др.
- **salary**: Предлагана заплата или диапазон на заплатата за дадената позиция.

Пакет "Interaction Management" (Управление на Взаимодействията):

Chat (Чат):

Атрибути:

- **chatID**: Уникален идентификатор на чата.
- **participants**: Списък с участници в чата.

Методи:

- **startChat()**: Започване на нов чат.
- **sendMessage()**: Изпращане на съобщение.

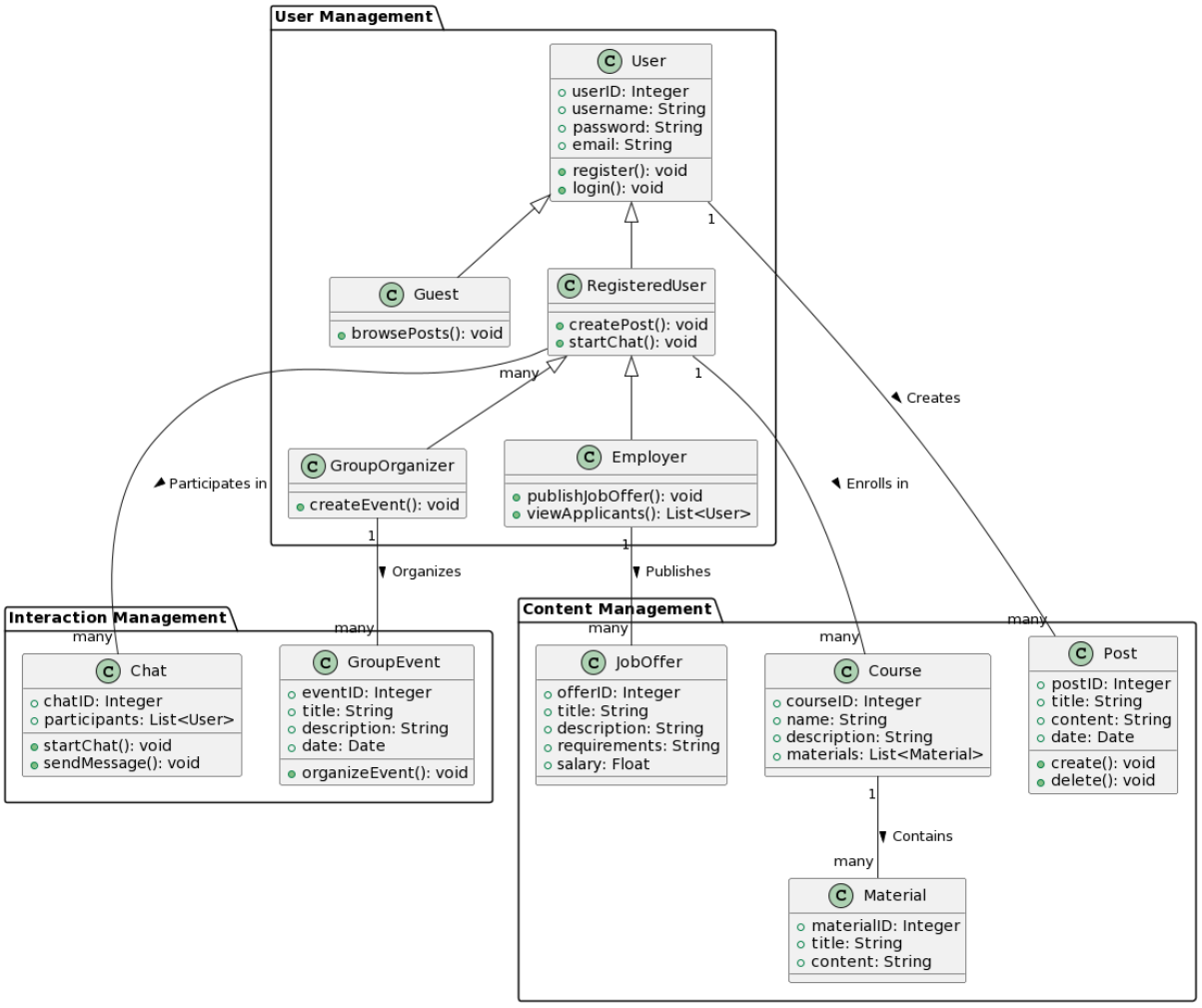
GroupEvent (Групово Събитие):

Атрибути:

- **eventID**: Уникален идентификатор на събитието.
- **title**: Заглавие на събитието.
- **description**: Описание на събитието.
- **date**: Дата на провеждане на събитието.

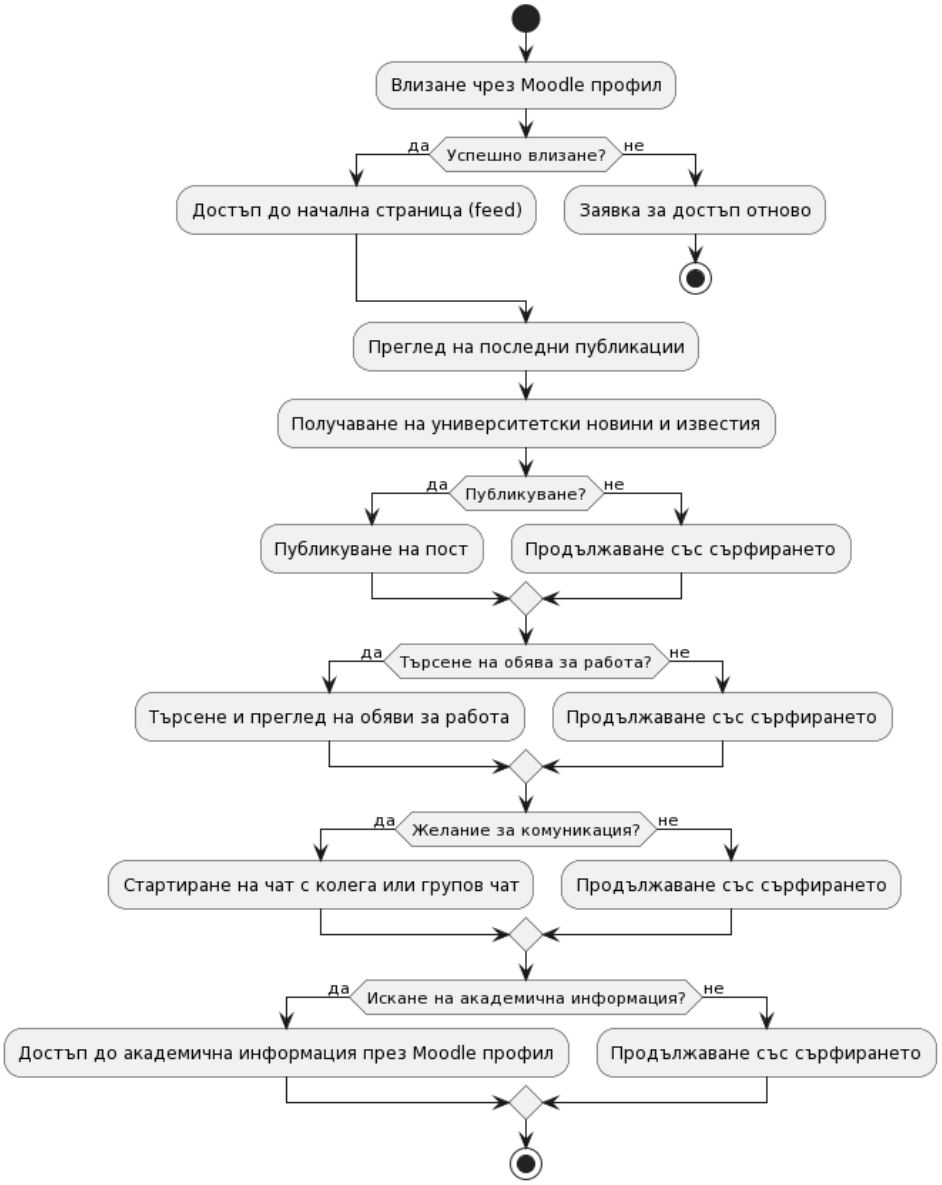
Методи:

- **organizeEvent()**: Организиране на групово събитие.



3.3. Процесен изглед

Процесният изглед на системата предоставя поглед върху ключовите бизнес процеси и потоци на действие в рамките на системата UniVerse. Той изобразява как функциите (описани в Use-Case изгледа) се превръщат в конкретни последователности от операции, интеракции и преходи. Чрез този изглед разработчиците и архитектите могат да разберат как потребителите ще взаимодействат с системата и как ще се осъществяват различните сценарии на употреба.



Влизане чрез Moodle профил: Първата стъпка в платформата "UniVerse" е влизането на потребителя чрез своя Moodle профил. Това гарантира, че всеки потребител е част от университетската общност и има право на достъп до платформата.

Достъп до начална страница (feed): След успешен вход, потребителят е пренасочен към своя личен feed, където може да вижда последни публикации, новини и обновления.

Публикуване на пост: Потребителят има възможност да сподели свои академични постижения, записки или други релевантни публикации.

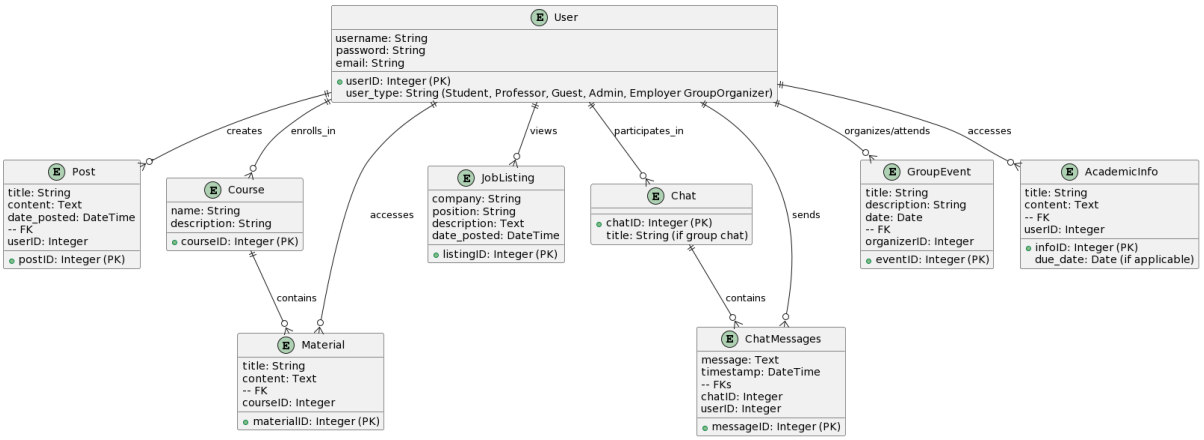
Търсене и преглед на обяви за работа: Платформата предоставя възможност на студентите да търсят работа, стаж или студентска практика.

Стартиране на чат с колега или групов чат: Потребителите могат да комуникират помежду си в реално време, което улеснява взаимодействието и обмена на информация.

Достъп до академична информация през Moodle профил: Чрез интеграцията с Moodle, потребителите имат възможност да получат академична информация, като предстоящи задания, изпити и оценки.

3.4. Изглед на данните

Базата данни на "UniVerse" е създадена, за да улесни комуникацията между различните компоненти на платформата и да осигури лесен достъп до нужната информация за всички актьори в системата. Нейната архитектура е проектирана така, че да отговори на нуждите за съвременна университетска социална платформа, като в същото време осигурява сигурност, надеждност и мащабируемост.



User (Потребител): Тази същност (entity) представлява всички потребители на платформата. Включва информация като уникален идентификатор, потребителско име, парола, имейл адрес и тип на потребителя (студент, преподавател, гост, администратор, работодател или групов организатор). Потребителите имат различни роли и права в системата.

Post (Публикация): Тази същност (entity) представлява публикации, които потребителите могат да създават. Включва информация като уникален идентификатор, заглавие, съдържание и дата на публикуване. Публикациите са свързани с потребителите, които ги създават.

Course (Курс) и **Material (Материал)**: Тези същности (entities) са свързани с академичната информация (AcademicInfo). Курсовете представляват учебни курсове, а материалите са свързани с тях. Курсовете имат информация като уникален идентификатор, име и описание. Материалите включват уникален идентификатор, заглавие и съдържание. Курсовете и материалите имат връзка помежду си.

JobListing (Обява за Работа): Тази същност (entity) представлява обяви за работа, които потребителите могат да разглеждат. Включва информация като уникален идентификатор, компания, позиция, описание и дата на публикуване.

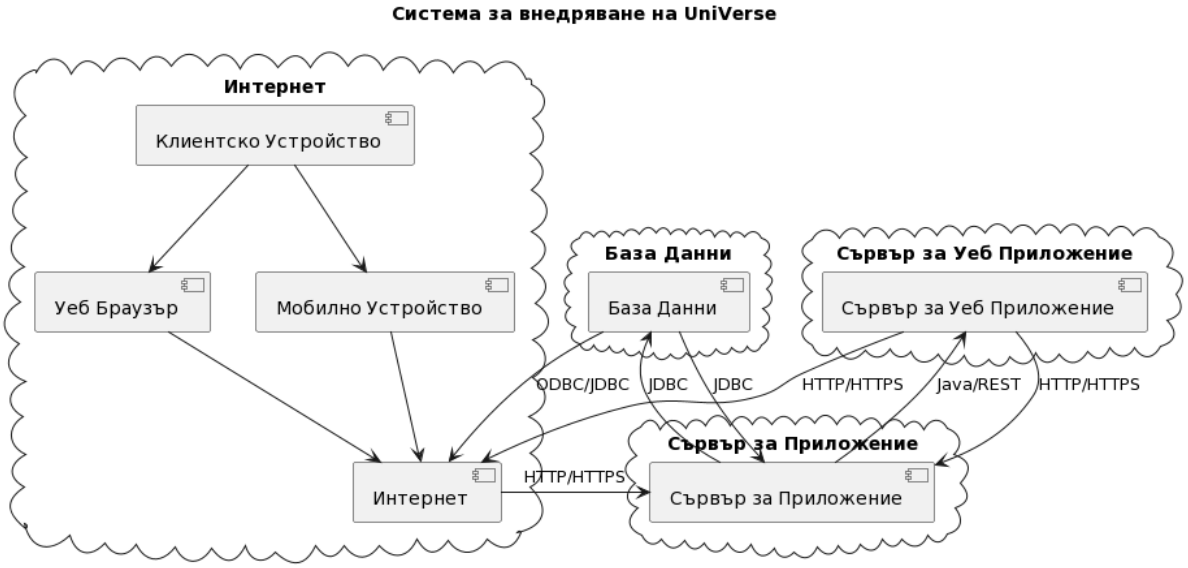
Chat (Чат) и **ChatMessages (Съобщения в Чата)**: Тези същности (entities) поддържат комуникацията между потребителите. Чатовете могат да бъдат индивидуални или групови. Съобщенията са свързани с определен чат и съдържат информация като уникален идентификатор, съобщение и времеви маркер.

GroupEvent (Групово Събитие): Тази същност (entity) представлява събития. Включва информация като уникален идентификатор, заглавие, описание и дата на събитието. Груповите събития могат да бъдат създадени и организирани от групови организатори.

3.5. Изглед на внедряването

Изгледът на внедряването представлява важна част от архитектурния дизайн на системата "UniVerse". Този изглед ни дава възможност да обобщим начина, по който различните компоненти и ресурси на системата са разположени и комуникират помежду си. Внедряването на софтуерните компоненти в сървъри и инфраструктура е ключов етап от процеса на разработка на системата и има съществено значение за нейната работоспособност и скалабилност.

Този изглед използва символи и символика, за да представи основните елементи на инфраструктурата, включително сървъри, бази данни, клиентски устройства и комуникационни канали. Чрез този анализ можем да разберем как различните компоненти взаимодействат помежду си, какви са начините на комуникация и къде се хостват софтуерните приложения.



Облак (Cloud): Използва се за представяне на физически или виртуални сървъри, където се хоства системата или приложението. Този символ обикновено представлява инфраструктурата, като уеб сървъри и сървъри за приложения.

Клиентско Устройство (Client Device): Представлява клиентски устройства като компютри, лаптопи, таблети и мобилни устройства, от които потребителите достъпват системата.

Сървър за Уеб Приложение (Web Application Server): Показва сървъра, който обработва HTTP/HTTPS заявки и предоставя интерфейса на системата чрез уеб браузърите.

Сървър за Приложение (Application Server): Представлява сървъра, който изпълнява приложението, включително логиката на приложението и връзката с базата данни.

База Данни (Database): Показва сървъра за база данни, където се съхраняват данните на системата.

Интернет (Internet): Представлява комуникацията чрез Интернет, която свързва всички компоненти на системата.

3.6. Изглед на имплементация

3.6.1. Концепция за слоеве:

3.6.1.1. Потребителски интерфейс

Този слой се отнася до всички елементи, които потребителите виждат и с които взаимодействат. Тук се включват уеб страници и други интерфейсни елементи. UI слоят предоставя интуитивен и удобен за потребителите на UniVerse дизайн.

3.6.1.2. Приложна логика

Този слой съдържа цялата приложна логика на UniVerse. Тук се извършват операции като обработка на заявки от потребителите, достъп до базата данни, изчисления и много други. Този слой е отговорен за бизнес логиката на системата.

3.6.1.3. Управление на база данни

Този слой управлява данните в базата данни на UniVerse. Тук се извършват операции като създаване, четене, актуализиране и изтриване на данни. Слоят е отговорен за съхранението и управлението на информацията.

3.6.1.4. Итреграция с Moodle

Този слой е специфичен за интеграцията с платформата Moodle. Тук се осъществява комуникацията между UniVerse и Moodle за автентикация на потребителите и обмен на информация.

3.6.2. Цел на слоевете:

3.6.2.1. Разделение на отговорности

Използването на слоеве позволява разделение на отговорности между различни части на системата. Този подход прави приложението по-структурирано и управляемо.

3.6.2.2. Съхранение на конфиденциална информация

Информацията, която трябва да бъде съхранявана в сигурност, може да бъде пазена в отделен слой, който има ограничен достъп.

3.6.3. Правила за реализация и употреба

3.6.3.1. Интерфейси

Всеки слой следва да има ясни интерфейсни връзки и да спазва стандартни правила за взаимодействие с другите слоеве. Това включва дефиниране на API и протоколи за комуникация.

3.6.3.2. Копия и възстановяване

Всеки слой трябва да има механизми за създаване на резервни копия на данните и възстановяване в случай на аварии или губене на данни.

3.6.3.3. Тестване и валидация

Всеки слой трябва да бъде тестван и валидиран за правилното си функциониране. Това включва използването на инструменти и методи за автоматизирано тестване, които да проверят, че всички функции работят коректно и отговарят на изискванията. Например, се използват модулни тестове за бекенда и UI/UX тестове за фронтенда.

4. Нефункционални изисквания

Нефункционалните изисквания за системата "UniVerse" играят важна роля при определянето на атрибутите на качеството и как те се реализират в архитектурата на приложението. В зависимост от реализацията на системата, могат да бъдат разгледани следните атрибути на качеството и свързаните с тях тактики:

4.1. Достъпност

UniVerse трябва да бъде налична за потребителите в рамките на 99.9% от времето. За постигане на този атрибут на качеството, системата използва надеждни сървъри и мрежови ресурси, с автоматично възстановяване при отказ и редовни бекъпи на данните.

4.2. Разширяемост

Изграждането на софтуерни системи, които са независимо разширяеми, е важно предизвикателство. Една независимо

разширяема система не само позволява на двама души да разработват независимо разширения към системата, но също така позволява двете разширения да бъдат комбинирани без глобална проверка на цялостта. Постигнато е чрез разделяне на отделните функционалности в модули, енкапсулирайки ги, което позволява да се добавят или премахват модули без да се афектира цялостната система.

4.3. Производителност

Системата поддържа стабилна производителност дори при натоварени периоди, когато стотици или хиляди потребители се възползват едновременно от UniVerse. Този атрибут на качеството се постига чрез оптимизирани заявки към базата данни и внимателно планиране на инфраструктурата.

4.4. Сигурност

Защитата на информацията и ограничаването на достъпа са важни аспекти на сигурността. Постига се чрез усилена аутентикация, управление на правата за достъп, криптиране на данни и регулярни одити за откриване на потенциални уязвимости. Добрата защита на данните предотвратява заплахи за сигурността като атаки чрез инжектиране, Cross-site scripting (XSS) атаки, атаки за удостоверяване и управление на сесии и др.

4.5. Възможност за тестване

Системата е проектирана и имплементирана с оглед на лесно и ефективно тестване. Примерни тестове са изолиране на модулите за тестване, използване на автоматизирани тестове и интегрирани инструменти за тестване. Освен това се използват добри и разбираеми конвенции за именуване на компоненти и файлове, и ясно регистриране на грешки, което улеснява значително тестването на софтуера.

4.6. Интероперабилност

За гарантиране на интероперабилност между различни системи, се прилагат стандартизирани протоколи и формати за обмен на данни. Спазването на стандартите позволява интегрирането на компоненти или услуги от различни доставчици или платформи. С други думи архитектурата е замислена да е отворена.

4.7. Използваемост

Лесната използваемост на системата се осигурява чрез удобен и интуитивен потребителски интерфейс, с подходящи за университетския контекст функционалности и поддръжка на добри практики за дизайн на потребителски изживявания, като консистентност, простота, адаптивност и достъпност.