MR. UNI

Курсова работа по Софтуерни Архитектури и Разработка на Софтуер

Изготвен от:
Мария Ташкова, 62294
Радо Димов, 62268

Съдържание

1. Въведение

1.1 Обща информация за текущия документ

1.1.1 Предназначение на документа

Документът има за цел да представи софтуерната архитектура на система, която подпомага работата на един университет – Mr.Uni.

1.1.2 Описание на използваните структури

1.1.2.1 Декомпозиция на модулите

Показва как са обособени логически свързаните модули на едно място. Системата е разпределена на отделни модули като основните такива са - User Interface, Server и DB. Всеки модул включва подмодули, които отговарят за съответните функционалности. Модулът UI представя визуално системата на потребителите. Server представя административните функции, бизнес логиката и осъществява връзката с останалите модули. DB съхранява и да предоставя достъп до данните в системата.

1.1.2.2 Структура на процесите

Дава по-ясна представа как се извършват процесите и в каква последователност се изпълняват действията. Показва кое действие в кой модул се осъществява и представя системата по по-разбираем начин за потребителите и заинтересованите лица.

1.1.2.3 Структура на внедряването

Представя по какъв начин са разпределени компонентите върху различните хардуерни и софтуерни системи. Също така показва някои от направените тактики с цел подобряване на качествените изисквания на системата. Като например употребата на две бази, множество сървъри, Load-balancing server и по какъв начин е осъществена комуникацията между тях.

1.1.3 Структура на документа

Въведение - секция 1

Декомпозиция на модулите - секция 2

- Общ вид на декомпозицията на модулите
- Контекстна диаграма
- Подробно описание
- Описание на възможните вариации

Структура на процесите - секция 3

- Първично представяне
- Описание на елементите и връзките
- Описание на обкръжението
- Описание на възможните вариации

Структура на внедряването - секция 4

- Първично представяне
- Описание на елементите и връзките
- Описание на обкръжението
- Описание на възможните вариации

Архитектурна обосновка - секция 5

Допълнителна информация - секция 6

1.2 Общи сведения за системата

Mr.Uni е система за управление на студентската информация, която ръководи процесите и финансовите операции в един университет. Уеб платформата предоставя възможност за комуникация между студенти, преподаватели и различните отделите в университета. Преподавателите и студентите притежават профили, в които се записват техните данни. Един студент може да осъществи операции свързани с преглед на студентска книжка, записване на избираеми дисциплини, получаване на различни видове справки, които съответно са защитени от фалшификация и да извършва плащания. Преподавателите имат възможност да изготвят учебната програма и да вписват оценки. Системата също има за цел да улесни работата на административните отдели в университета, като им позволява да извършат своите задължения по лесен и удобен начин.

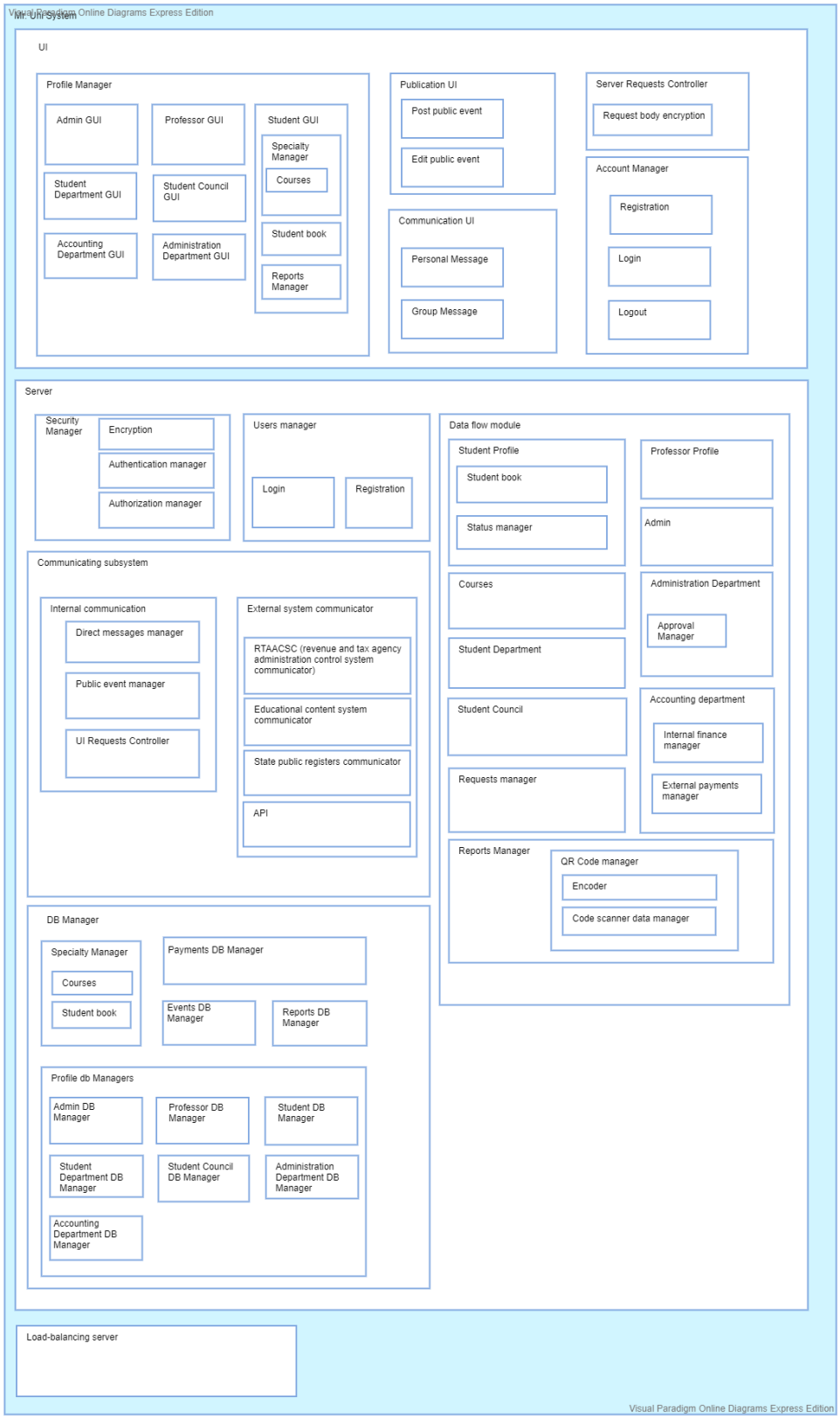
1.3 Терминологичен речник

- **Активен излишък** (Active redundancy, hot restart) – е софтуерна тактика за изправност, при която важните компоненти на системата се дублират. Те се поддържат в едно и също състояние и приемат един и същи вход и извършват една и съща работа едновременно.
- **База от данни** - колекция от информация, която е така организирана, че да може лесно да се достъпва, управлява и актуализира
- **Декриптиране** – процес на нормално прилагане на криптографско преобразуване на шифриран текст в открит.
- **Интерфейс** (interface) – споделена граница, между която два отделни компонента на компютърна система си обменят информация
- **Криптиране** - процесът на кодиране на информацията по начин, който предотвратява достъпа на неоторизирани лица до нея.

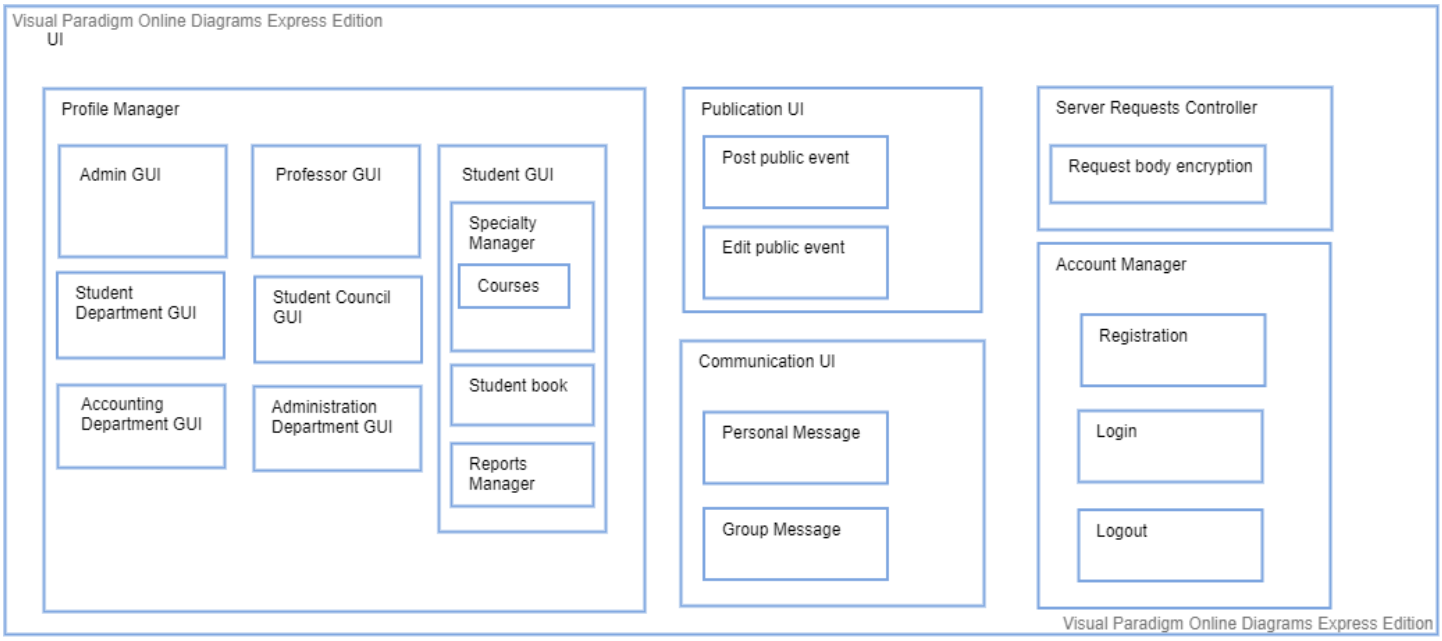
- **Ограничаване на комуникацията** (Communication constraint) - е тактика за изменяемост, при която се ограничава броя на модулите, с които даден модул комуникира. Също се ограничава и информацията, която се предава.
- **Пасивен излишък** (Passive redundancy) - е софтуерна тактика за изправност, при която един от компонентите реагира на събитията, информира останалите компоненти. При дефект, се сменя активният компонент.
- **Режим на сянка** (Shadow mode) – е тактика за изправност, при която поправен модул работи паралелно в системата. Не се въвежда в реална употреба, само се тества, докато не се установи, че работи коректно.
- **API** (Application Programming Interface) - е интерфейсът на изходния код, който операционната система или нейните библиотеки от ниско ниво предлагат за поддръжката на заявките от приложния софтуер или компютърните програми.
- **Cloud** - са компютърни услуги, предоставяни на потребител чрез отдалечен компютър, към който потребителят се свързва чрез интернет или чрез специална комуникационна линия.
- **Controller** - се нарича управляващо устройство, което контролира и регулира определени процеси.
- **GUI** (Graphical user interface) - е разновидност на потребителски интерфейс, в който елементите, предоставени на потребителя за управление, са изпълнени във вид на графични изображения
- **HTTP protocol** (Hypertext Transfer Protocol) - е мрежов протокол, от приложния слой на OSI модела, за пренос на информация в компютърни мрежи.
    - **Status 300** (Multiple Choices) - код за отговор на състояние, показва, че заявката има повече от един възможен отговор.
- **QR Code** (Quick Response) - е специфичен матричен баркод, разпознаваем от специални QR четци за баркодове или камери на мобилни телефони. Баркодът се състои от черни модули, подредени в квадратен шаблон върху бял фон.
- **Server** - стартирана инстанция на софтуерна система, която може да приема заявки от клиент и да връща подходящи отговори.
- **TCP protocol** (Transmission Control Protocol) - е мрежов протокол за управление на обмена на информация, един от основните, използвани в интернет. Използвайки го приложенията в мрежата могат да създават връзки едно с друго и чрез тях да обменят данни в пакети.
- **User Interface** - потребителски интерфейс, е мястото за взаимодействие между човека и машината с цел да се позволи ефективно управление и контрол върху машината от страна на човека, като в същото време машината връща обратно информация, която подпомага процеса на вземане на решения от оператора.

2. Декомпозиция на модулите

2.1 Общ вид на декомпозиция на модулите на системата



2.2 User Interface



2.2.1 Предназначение на модула

Модулът UI осигурява видим достъп на различните потребители до системата като те се разделят на 7 основни типа - администратор, студент, преподавател, учебен отдел, счетоводен отдел, студентски съвет, административен отдел. Целта му е да позволи използване на специфични функционалности за съответният тип потребител.

2.2.2 Основни отговорности на модула в системата

Модулът предоставя различни типове информация по достъпен за потребителя начин и осъществява комуникацията на потребителя с модул Server.

2.2.3 Описание на интерфейсите на модула

2.2.3.1 Account Manager

Осъществява връзката на външни за системата лица до функционалността, която ги автентикира и оторизира.

2.2.3.1.1 Registration

Позволява на гостите на системата да попълнят необходимата информация и ги изпраща до модул Server, където се случва самата регистрация и валидация на въведените данни.

2.2.3.1.2 Login

Има за цел да идентифицира външните лица, като изпраща информацията до Server-а, където се случва проверката.

2.2.3.1.3 Logout

Когато потребителят иска да излезе от текущата сесия, в която е логнат.

2.2.3.2 Communication UI

Целта му е да визуализира процеса по изпращане на съобщение. Самото изпращане се осъществява в Server-а**.** Тази функционалност е достъпна до всички типове потребители.

2.2.3.2.1. Personal Message

Личното съобщение може да се изпраща директно към посочен потребител, след като се установи, че профилът му е валиден.

2.2.3.2.2 Group Message

Груповото съобщение позволява да се изпрати съобщение на група от потребители.

2.2.3.3 Publication UI

Системата поддържа механизъм за публикуване на публични събития. Като тази функционалност е достъпна за всички потребители. Целта е да визуализира информацията относно събитието, което ще бъде публикувано и да го препрати към Server-а.

2.2.3.3.1 Post public event

Позволява на потребителите да заявят, че желаят да публикуват събитие.

2.2.3.3.2. Edit public event

Позволява извършване на промяна на публикацията относно събитието.

2.2.3.4 Profile Manager

Модулът е достъпен след като самоличността на потребителя е установена. Позволява разделянето на отделните видове потребители. Като всеки отделен тип потребител има специфични функционалности и правомощия, които го отличават от останалите потребители. Като даден потребител се идентифицира в съответния модул, получава достъп до функционалностите, които са му необходими.

2.2.3.4.1. Admin GUI

Модулът служи за визуално представяне на правомощията и функциите характерни за администратора.

2.2.3.4.2. Professor GUI

Модулът служи за визуално представяне на правомощията и функциите характерни за потребител от тип - преподавател.

2.2.3.4.3. Student GUI

Модулът служи за визуално представяне на правомощията и функциите характерни за потребител от тип - студент.

2.2.3.4.3.1. Specialty Manager

Модулът има за цел да визуализира характерните за съответната специалност дисциплини, разпис, хорариум и др.

2.2.3.4.3.1.1. Courses

Студентите имат право да избират, кои курсове да запишат, ако те са включени в специалността им.

2.2.3.4.3.2. Student book

Всеки студент притежава студентска книжка. И всеки студент има право да я преглежда.

2.2.3.4.3.3. Reports Manager

Студентите могат да генерират различни видове официални справки за студентския им статус. Модулът има за цел да визуализира съответната справка, поискана от студента.

2.2.3.4.4. Student Department GUI

Модулът служи за визуално представяне на правомощията и функциите характерни за потребител от тип - учебен отдел.

2.2.3.4.5. Student Council GUI

Модулът служи за визуално представяне на правомощията и функциите характерни за потребител от тип - студентски съвет.

2.2.3.4.6. Accounting Department GUI

Модулът служи за визуално представяне на правомощията и функциите характерни за потребител от тип - счетоводен отдел.

2.2.3.4.7 Administration Department GUI

Модулът служи за визуално представяне на правомощията и функциите характерни за потребител от тип - административен отдел.

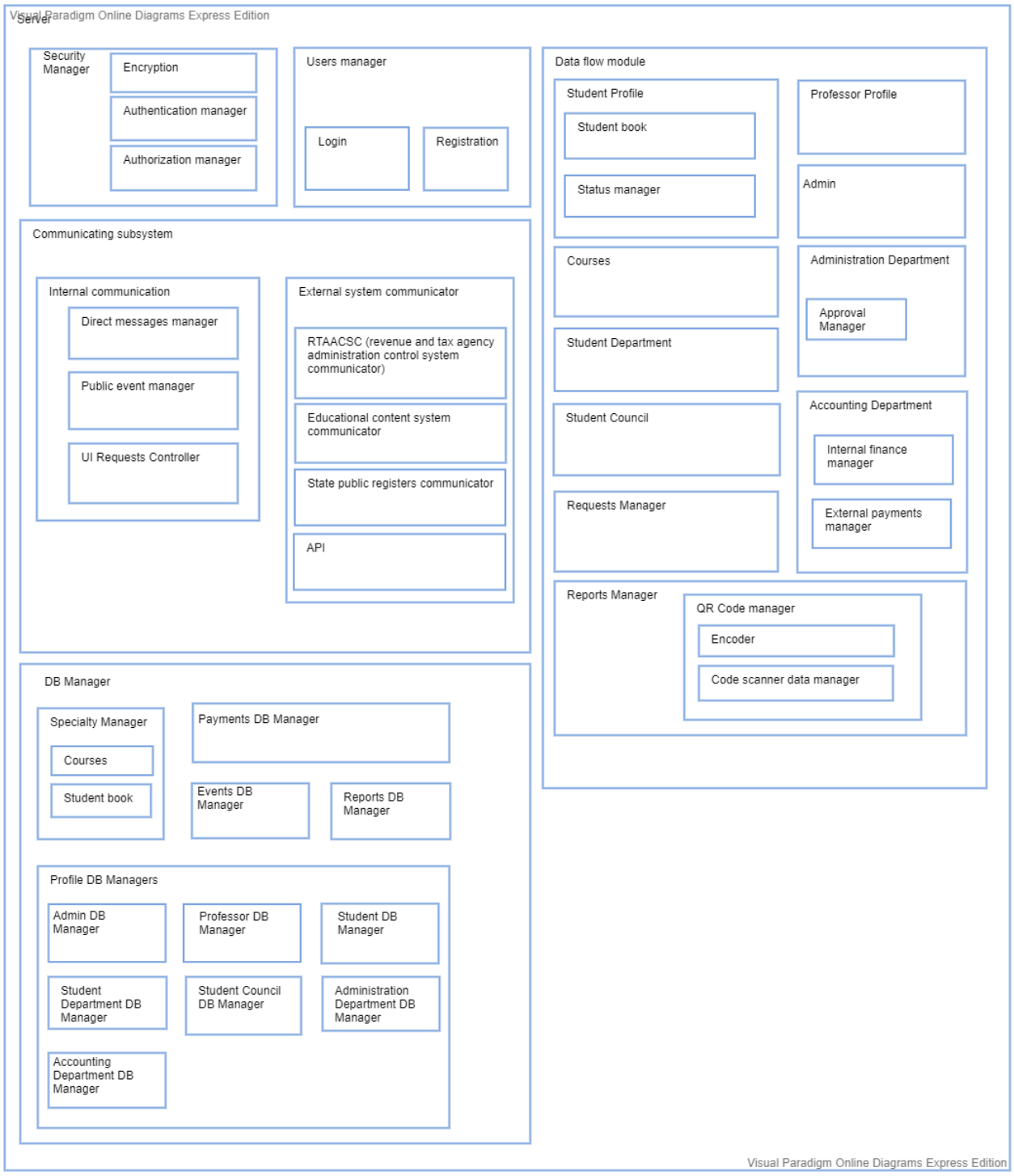
2.2.3.5 Server Request Controller

Отговаря за комуникацията със сървъра. Приема и обработва заявки и изпраща отговори.

2.2.3.5.1 Request body encryption

Отговаря за криптиране и декриптиране на sensitive информация, която се изпраща към и се получава от Server-а.

2.3 Server



2.3.1 Предназначение и основни отговорности на модула в системата

Модулът Server отговаря за бизнес логиката, по всички потребителски и системни операции, които съдържа Mr. Uni.

2.3.2 Описание на интерфейсите на модула

2.3.2.1 Security Manager

2.3.2.1.1 Encryption

Отговаря за криптиране и декриптиране на sensitive информация, която се изпраща към и се получава от UI-а. Също така криптира личните съобщения между потребителите на системата и ги декриптира при получаване.

public string Encrypt(string password)
public string Decrypt(string encryptedPassword)

2.3.2.1.2 Authorization manager

Отговаря за това да проверява дали потребителя има права за да извърши определено действие в системата.

2.3.2.1.3 Authentication manager

Отговаря за идентификацията на потребителя. Свързва се с Register и Login в User Manager.

public bool ValidateUserCredentials(User user) – при липсващи елементи в user обекта, се хвърля AuthenticationException

2.3.2.2 User manager

Управлява логиката зад създаването и влизането в даден профил, като приема информация изпратена от UI и след извършването на определени валидации в Authentication изпраща нужните данни и инструкции към Database Manager-а. Работата се разделя съответно между:

2.3.2.2.1 Login manager

Login manager-ът стои зад оформянето на информация въведена от потребителя и се свързва с Authentication, за да се извърши съответните валидации за автентикация.

public string GetUsername()
public string GetPassword()
public void DoLoginRequest ()

2.3.2.2.2 Registration manager

Registration manager стои зад оформянето на информация въведена при регистрация, като прави нужните валидации, за вече съществуващи такива потребителски имена в системата

public string GetUsername()
public string GetPassword()
public string GetEmail()
public void DoRegisterRequest ()

2.3.2.3 Communication subsystem

2.3.2.3.1 Internal communication

Отговаря за комуникацията, която се осъществява вътре в самата система.

2.3.2.3.1.1 Direct messages manager

Представлява модул контролиращ личните съобщения между потребителите.

2.3.2.3.1.2 Public events manager

Отговаря за създаването и промените по публикациите на публичните събития.

2.3.2.3.1.3 UI request controller

Модул за комуникация с UI модула откъдето пристигат заявки. Определя към кои модули да се препращат съответно различни отговори в зависимост от обработените данни в Server модула. Модулът използва модула Encryption в Security Manager при нужда.

2.3.2.3.2 External system communicator

Включва подмодули, които отговарят за комуникацията на системата с външни системи.

2.3.2.3.2.1 API

Осъществява начин чрез, който другите външни системи могат да използват нашата. Обработва и отговаря на заявки на чужди системи, които използват нашата.

2.3.2.3.2.2 RTASC (revenue and tax agency system communicator)

Нашата система използва дадена информация от и си комуникира със системата за контрол на национална агенция за приходите и данъчната администрация. за специфичната комуникация с тази система отговаря модулът RTASC.

2.3.2.3.2.3 Educational content system communicator

Отговаря за специфичната комуникация с дадена система за управление на учебното съдържание, като оформя получената информация и спомага за нужните операции с нея.

2.3.2.3.2.4 State public registers communicator

Отговаря за специфичната комуникация с държавните публични регистри, като оформя получената информация и спомага за нужните операции с нея.

2.3.2.4 Data flow module

Този модул служи за организиране на информацията и операциите по потребителските данни в сървъра. Тоест той се използва, за да съдържа класове, с които да се организират част от функциите и данните за изпълнение на бизнес логиката.

2.3.2.4.1 Reports Manager

Отговаря за изготвянето и управлението на справките в системата.

2.3.2.4.1.1 QR Code manager

Този подмодул на Reports manager отговаря за QR Code-а, с който ще осигури верифициране на хартиеното копие на справка с електронния ѝ вариант.

2.3.2.4.1.1.1 Encoder

Генерира различни QR кодове, които ще се прикрепят към хартиеното копие на справката.

2.3.2.4.1.1.2 Code scanner data manager

Обработва снимка на QR Code и сравнява с дадена справка, за да се уверим, че даден документ е истински.

2.3.2.4.2 Requests Manager

Управлява предложенията за учебни планове, както и всякакви заявки от потребителите включително споменатите в изискванията по системата.

2.3.2.4.3 Student Profile

Оформя информацията за студентите в класове и обекти и техни функции, за да се осъществят по-лесно нужните операции по бизнес логиката в сървъра.

2.3.2.4.3.1 Student book

Модулът отговаря за организиране на информацията по студентските книжки в класове и обекти и техни функции, за да се осъществят по-лесно нужните операции по бизнес логиката. Оценките в книжката се попълват от съответният преподавател, който води дисциплината. Студентът има право да прегледа студентската книжка.

2.3.2.4.3.2 Status manager

Модулът отговаря за организиране на информацията по студентските статуси в класове и обекти и техни функции с цел да се осъществят по-лесно нужните операции по бизнес логиката.

2.3.2.4.4 Professor Profile

Оформя информацията за преподавателите в класове и обекти и техни функции, за да се осъществят по-лесно нужните операции по бизнес логиката в сървъра.

2.3.2.4.5 Admin

Оформя информацията за администратори в класове и обекти и техни специфични права, за да се осъществят по-лесно нужните операции по бизнес логиката в сървъра.

2.3.2.4.6 Student Department

Оформя информацията за потребителите, които са от тип студентски отдел в класове и обекти и техни функции, за да се осъществи по-лесно нужната функционалност по бизнес логиката в сървъра.

2.3.2.4.7 Student Council

Оформя допълнителната информация за студентите, които са от студентски съвет в класове, за да се осъществи нужната функционалност по специфичните права, които имат дадените членове на съвета.

2.3.2.4.8 Accounting Department

Оформя нужната информация за потребителите от финансовия отдел, за да може след това да се извършват по по-лесен и удобен начин операциите свързани с плащания в системата.

2.3.2.4.8.1 Internal finance manager

Контролира финансовите операции, засягащи другите потребители.

2.3.2.4.8.2 External payments manager

Отговаря за разплащанията с външни изпълнители на услуги.

2.3.2.4.9 Courses

Оформя информацията за курсовете в класове и обекти и техни функции, за да се осъществят по-лесно нужните операции по бизнес логиката в сървъра. Това включва преглед на налични курсове, записване на курс и проверка за необходими компетентности.

2.3.2.4.10 Administration Department

Модулът служи за отделяне на правомощията и функциите характерни за типа потребител - административен отдел. Като помага по-лесно да се осъществят операциите, за които този отдел отговаря.

2.3.2.4.10.1 Approval manager

Като подмодул на Administration Department има за цел да улесни работата на модула. Обработва предложенията за учебни планове (специалности) и програми (курсове) и кои от тях ще бъдат приети от административния отдел.

2.3.2.5 DataBase Manager

2.3.2.5.1 Предназначение на модула

Модулът Database Manager има за цел да управлява постъпването и извеждането на информация от базата данни. Работата му е да достъпва базата, да чете необходимата информация и я взема от там.

2.3.2.5.2 Основни отговорности на модула в системата

Модулът изолира базата, което осигурява още едно ниво на сигурност при достъпа до базата данни. Също така разпределя заявките между отделните инстанции на базата и осигурява по четимо (достъпно) представяне на информацията за потребителя.

2.3.2.5.2.1 Specialty Manager

Модулът има за цел да разграничи, различните атрибути свързани със специалността на студента, за да се улесни работата на базата, в която се съхранява информацията.

2.3.2.5.2.1.1 Courses

Отговаря за оформяне на информацията свързана с курсовете, които студентът в рамките на неговата специалност ще има право да запише, при условие, че профилът му отговаря на входните изисквания за компетентности за съответния курс.

2.3.2.5.2.1.2 Student book

Отговаря за оформяне на информацията свързана с оценките на студента.

2.3.2.5.2.2 Events DB Manager

Модулът има за цел да разграничи и да оформи, различните атрибути свързани с информацията получена от публичните събития (еднократни курсове, състезания, събирания на групи по интереси и т.н.) и съответно тази информация ще се съхранява в базата.

2.3.2.5.2.3 Payments DB Manager

Отговаря за цел улесни работата на базата, като разграничи и оформи информацията за всички плащания извършени в системата - студентски такси, възнаграждения на служителите и тн.

2.3.2.5.2.4 Reports DB Manager

Модулът отговаря за справките и съответно информацията свързана с тях да бъде предоставена на базата.

2.3.2.5.2.5 Profile DB Manager

Отделните типове потребители, се нуждаят от различна информация за да работят по предназначение. Следователно модулът цели да ги разграничи, което значително ще улесни работата на базата. Ако даден потребител се идентифицира в съответния модул, получава достъп до функционалностите, които са му необходими.

2.3.2.5.2.5.1 Admin DB Manager

Осигурява на базата съответната информация, която може да извлече от този тип потребителски акаунт.

2.3.2.5.2.5.2 Professor DB Manager

Осигурява на базата съответната информация, която може да се извлече от типа потребител - преподавател. В базата се записват техните данни, както и информация за техните компетентности - кои специалности водят, курсове.

2.3.2.5.2.5.3 Student DB Manager

Осигурява на базата съответната информация, която може да се извлече от типа потребител - студент. В базата се записват техните данни, както и информация за компетентности им - притежават студентска книжка, водят се към определена специалност, одобрени курсове.

2.3.2.5.2.5.4 Student Department DB Manager

Осигурява на базата съответната информация, която може да се извлече от типа потребител - учебен отдел. Включва приемане на предложения за учебни планове (специалности) и програми (курсове).

2.3.2.5.2.5.5 Student Council DB Manager

Притежава правомощията и функциите характерни за типа потребител - студентски съвет. И осигурява на базата съответната информация, която може да извлече от този тип потребителски акаунт.

2.3.2.5.2.5.6 Administration Department DB Manager

Осигурява на базата съответната информация, която може да се извлече от типа потребител - административен отдел. Това включва одобряване на предложения за учебни планове (специалности) и програми (курсове).

2.3.2.5.2.5.7 Accounting Department DB Manager

Осигурява на базата съответната информация, която може да се извлече от типа потребител - счетоводен отдел. Тоест съдържа информация за финансовите операции, засягащи другите потребители.

2.4 Load-balancing server

Справя се с пикови натоварвания при множество потребителски заявки като разпределя заявките между множество сървъри, за да избегне забавяне при потока на данни при бизнес логиката.

3. Описание на допълнителните структури

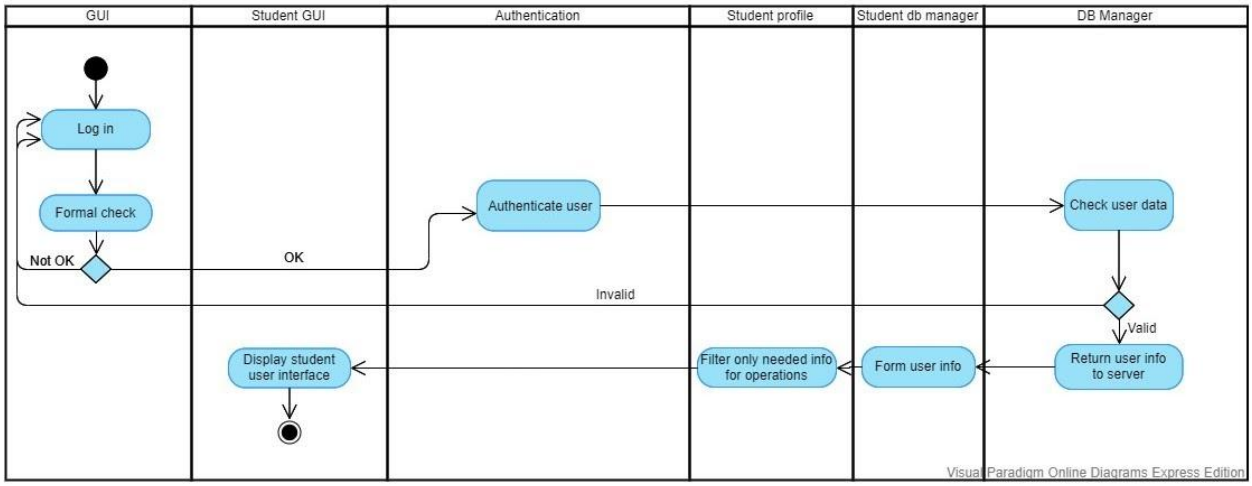
3.1 Структура на процесите

3.3.1 Мотивация на избор

Диаграмата има за цел да изобрази някои функционалности, които Mr.Uni изпълнява. Съсредоточихме се точно върху тази структура, защото заинтересованите лица могат по-лесно да добият представа как работи системата и в каква последователност се извършват отделните процеси. Дава по-ясна представа кой процес в коя част на системата се изпълнява, кога започва, и кога приключва и какви са възможните изходи от него. Това със сигурност ще допринесе за да постигнем висока производителност и бързодействие.

3.3.2 Първично представяне

Диаграмата представя процес за идентификация на потребител от тип студент.



3.3.3 Описание на елементите и връзките:

Целта на диаграмата е да покаже какви са отделните процеси, които протичат при вход на един посетител. При вход се извършва проверка на въведените от потребителя данни, ако е въвел некоректни данни съответно се връща от начало за да направи нов опит. Ако са въведени валидни символи, проверката продължава на следващ етап – автентикация в Authentication, където данните преминават през модула DB Manager с проверка в DB дали тези потребителски данни съществуват. Ако тези данни не съществуват, посетителят се връща към стъпката за вход. Ако данните са валидни следва извличане на нужните данни на потребителя от базата, формиране на обекти, чрез които борави сървъра и съответно получава достъп до

характерните за този тип акаунт функционалности. Накрая потребителят се препраща на страница, като му се визуализират функционалности, които съответстват на акаунта му. В случая сме представили как се осъществява идентификацията на потребител от тип Student.

3.3.4 Описание на обкръжението

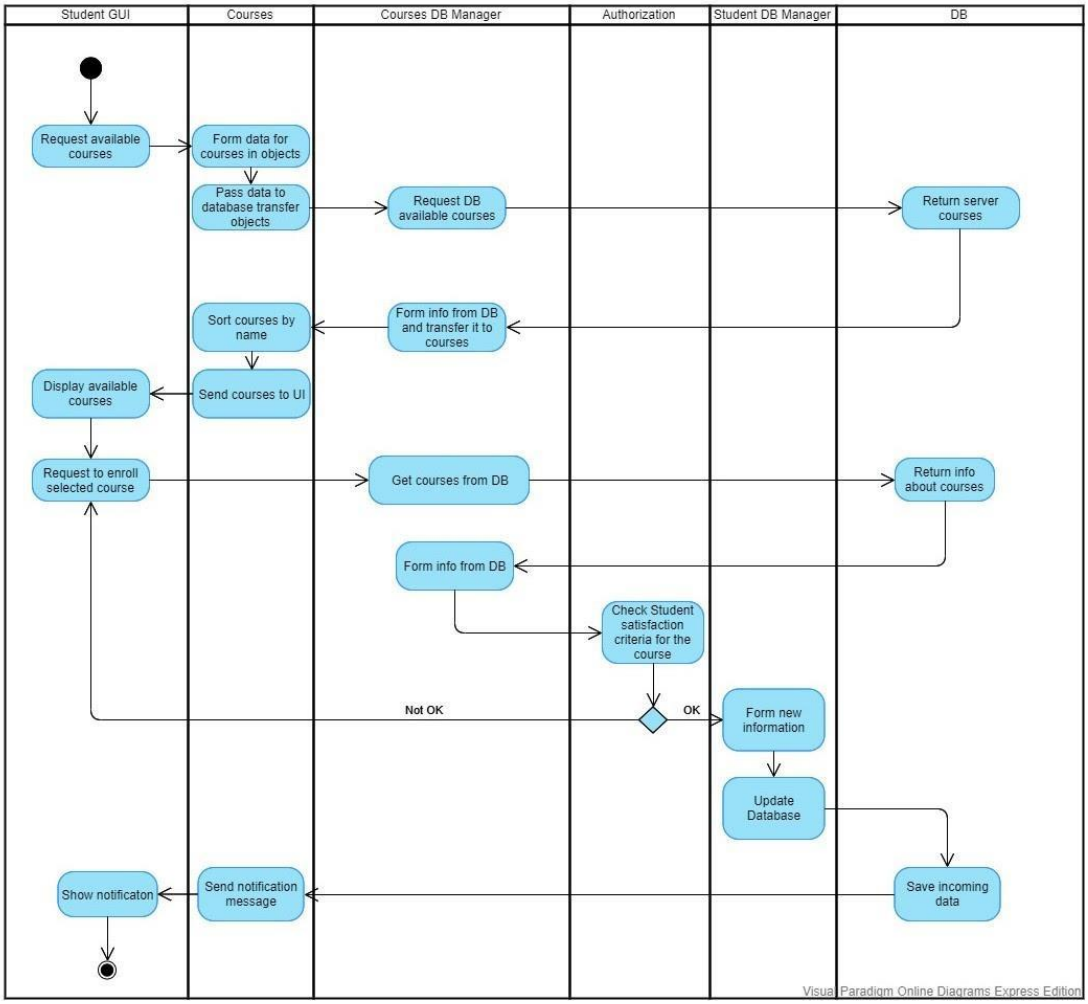
Входът на един потребител се извършва през Интернет, като минава през различни валидации и проверки.

3.3.5 Описание на възможните вариации

Възможна е потребителят да бъде информиран при въвеждане на некоректни данни, като се специфицира каква точно е възникналата грешка.

3.3.6 Първично представяне

Диаграмата представя процес за записване на одобрени курсове от студент.



3.3.7 Описание на елементите и връзките:

Потребителят от тип Student прави заявка, че желае да разгледа достъпните курсове. Информацията преминава през Courses, където се формират класове, с които да се прехвърля информацията и да се извършват нужните операции в сървъра. След това информацията се подава на Courses DB manager, където се извършва допитване към базата данни. Базата връща списъкът с курсовете и модулът Courses DB Manager ги оформя в обекти от езика. След това информацията от обектите се подава на Courses където курсовете се сортират и след това се изпращат към UI, където ще се визуализират на екрана. Студентът си избира име на курс за записване. Информацията за курса се извлича от базата и в Authorization следват процеси по установяване на правата, които студентът има, компетентности за курса и дали студентът задоволява изискванията за съответния курс. Ако проверката за оторизация е неуспешна, на студентът му се дава повторен опит да избере друг курс. Ако проверката за оторизация е успешна информацията за студента и исканият курс за записване се обновяват в Student DB manager и базата се достъпва, за да се запазят промените. В Server записва желаните дисциплините и след като информацията е преработена от модула DB Manager, новите данни се записват в базата. През отделните модули в сървъра, през които преминава информацията се връща сигнал за успешна операция и накрая към UI се изпраща отговор за успешно извършена заявка за записване на курс. Съобщението се визуализира в Student GUI. Така завършва процесът.

3.3.8 Описание на обкръжението

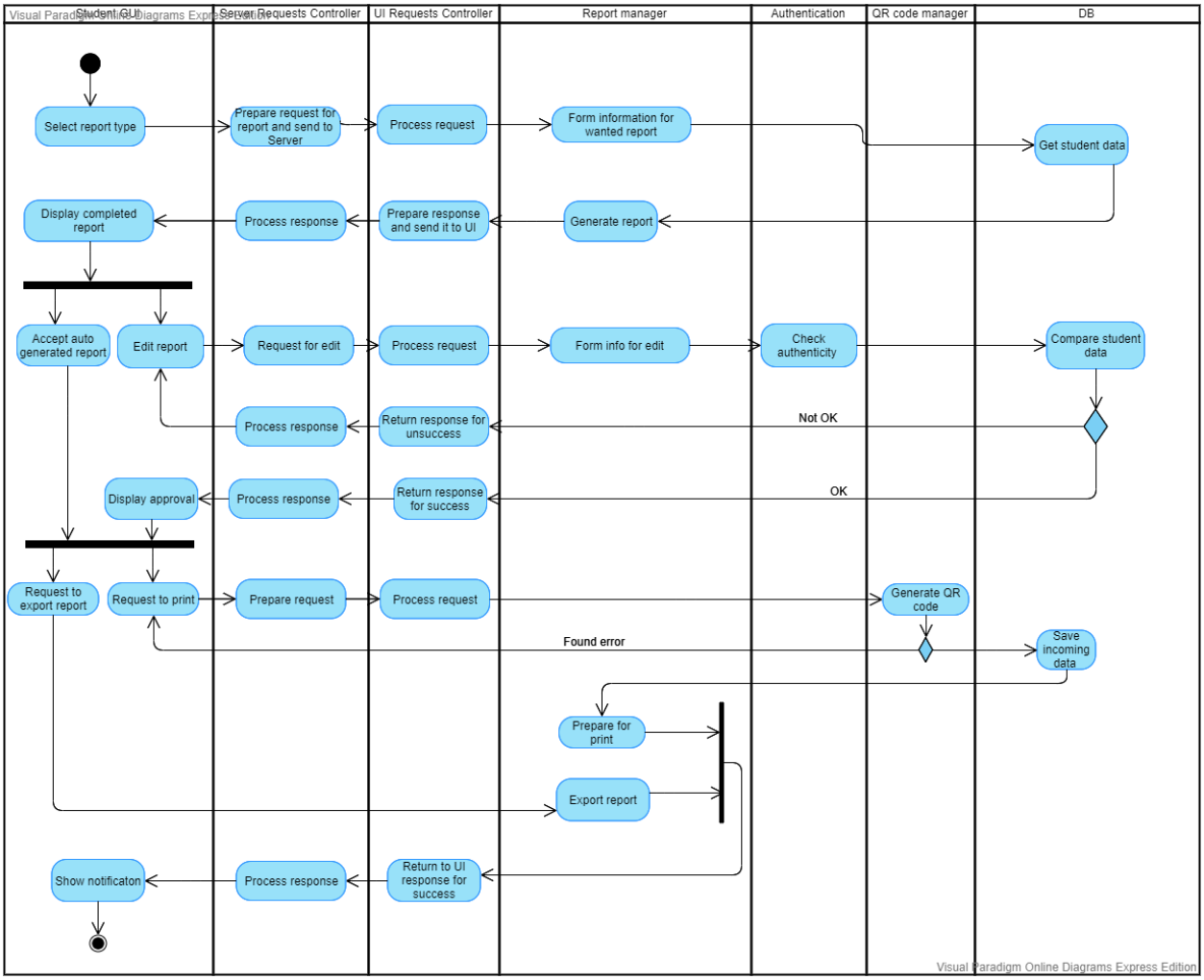
Достъпването на системата от потребители, включително и този процес, стават през браузър.

3.3.9 Описание на възможните вариации

Друг вариант е още при показване на курсовете да излизат само тези, за които даденият студент покрива критериите. По този начин няма да има възможност да избере нещо, за което няма правата.

3.3.10 Първично представяне

Диаграмата представя процес за генериране на различни видове официални справки относно студентския статус на студент.



3.3.11 Описание на елементите и връзките:

Потребителят от тип Student избира вида на справката, която желае да получи. През целият процес връзката между модулите на UI и модулите на Server се осъществява чрез Server Requests Controller и UI Requests Controller. Информацията, която се иска в потребителската заявка се оформя в Report Manager, за да може модулите в Server да работят с нея. Всяка комуникация с базата се осъществява, чрез подмодули на DB Manager. Тъй като е подобно при всеки процес в тази диаграма не сме го повторили. През Report DB manager се осъществява връзката с базата и се взима необходимата информация за попълване на желаната справка. В Report manager се автогенерира справката и се визуализира попълнена в Student GUI. Студентът има право да избере дали да редактира справката или да одобри автогенерираните данни. Ако реши да прави промени, се извършва проверка в Authentication с цел да не се фалшифицират данните. След като данните са попълнение или одобрени успешно, студентът може да си разпечата справката или да я запази в електронен вариант. В случай, че реши да я разпечата, хартиеното копие получава QR code, чрез който се прави

верификация с електронният вариант на справката, който се генерира в QR code manager. При възникване на грешка се връща response за неуспех към UI и се дава отново възможност на студента да заяви принтиране. При успех тази информация се запазва в базата. Когато студентът успее да запази справката или да я разпечата, получава съобщение за успешно завършен процес.

3.3.12 Описание на обкръжението

Процесът по принтиране, който ще последва приготвяне за принтиране се свързва с дадена система за принтиране.

3.3.13 Описание на възможните вариации

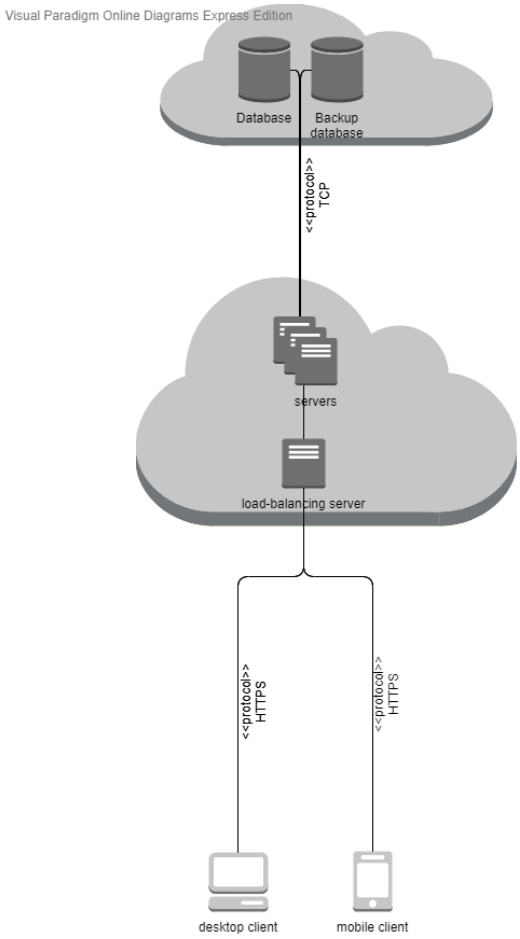
Възможна вариация на механизъм за верифициране с електронният вариант на справката, вместо QR code, може да е с подпис, като ще трябва да се добави модул и UI за подписи.

3.2 Структура на внедряването

3.2.1 Мотивация на избор

Deployment е ключова структура за MR. Uni. В декомпозицията на модулите всеки отделен главен модул е на отделно устройство. Някои модули може да са дублирани на няколко различни машини (сървъри).

3.2.2 Първично представяне



3.2.3 Описание на елементите и връзките:

- Desktop client, mobile client: Различните примери за клиенти, които достъпват системата.
- HTTPS protocol: Клиентската част си комуникира със сървъра, чрез HTTPS протокол за комуникация. Идеята зад HTTPS е, че това е сигурна връзка, която ще предотврати опитите на външни лица за нарушение на комуникацията между нашият сървър и потребителските интерфейси на нашите потребители.
- Load-balancing server: Едно от изискванията към системата е това тя да бъде достъпна 24/7. За да спазим това изискване използваме отделен сървър, който да пренасочва клиентските заявки към сървъри на различни машини. По този начин при голямо натоварване бизнес логиката, която обслужва различните клиенти няма да се изпълнява на една машина, а на много и така всичко ще върви по-бързо. Вътрешно load-balancing server работи като при получаване на заявка, той я пренасочва към един от ip адресите на задните сървъри, а към клиентът първо се връща заявка от тип 300.

- Servers: Сървърите в системата съдържат бизнес логиката. Всеки сървър е на отделна машина. Те комуникират с load-balancing server-а, клиентите на системата, както и базата данни. Получават и обработват заявки от клиентите и разпределителя на товара и връщат отговори.
- Database: Базата данни съдържа цялата информация нужна за съхранение в системата.
- Clouds: Load-balancing server, servers и database са на облаци, като целта е да излезе по-евтина поддръжката на системата, тъй като ресурсите в един облак се използват много по-ефективно и няма опасност от излишно заплатени неизползвани процесорни запаси.
- Backup database – Тъй като едно от изискванията ни е системата да работи 24/7, сме използвали тактика за да осигурим изправност на системата - активен излишък. Втора база данни, която ще се активира и системата ще използва, ако нещо се случи с първата.

2.3.1 Описание на обкръжението

Комуникацията на потребителите със системата се извършва чрез браузър, който съответно не е част от системата.

2.3.2 Описание на възможните вариации

Базите с данни и сървъра да не са на облак. Тогава възможен вариант е да се закупят или наемат частни машини, които да се използват от системата като се държат сървърите и базите с данни на тях.

4. Архитектурна обосновка

1. **Системата обслужва следните отдели в университета:**
    a. **Учебен отдел**
    b. **Счетоводен отдел**
    c. **Студентски съвет**
    d. **Административен отдел**

Четирите отдела се реализират чрез Profile Manager в DB Manager, Profile Manager в UI и Data flow module в Server като съответно за всеки отдел има модули, чрез които се оформя комуникацията с базата, визуализацията в UI и потока на данни между тях. Всички тези модули манипулират информацията от извършени операции от дадените потребители и оформят потокът на данни.

2. **Всеки отдел предполага наличието на определен тип потребители. Освен това съществуват и администратори на системата, преподаватели и студенти.**

Пояснява изискване номер 1. и го допълва. Тъй като има различни видове потребители, това предполага, че те имат различни права, следователно в системата се прилагат тактики за сигурност от тип устояване на атаки, а именно чрез автентикация и оторизация, които са осъществени в Server, като подмодули на Security Manager.

3. **Потребителите от учебен отдел приемат предложения за учебни планове и програми от преподавателите.**

4. **Предложенията за учебни планове (специалности) и програми (курсове) се одобряват от административния отдел**

Осъществени са чрез модула Request Manager. Като в него фигурира цялата информация свързана с предложенията, както и операциите на различните потребители по предложенията се извършват вътре.

5. **Системата поддържа профили на студентите и преподавателите, в които се записват техните данни, както и информация за техните компетентности.**

Professor Profile и Student Profile се грижат, както за потока на данни произлязъл от извършените операции от дадените потребители, така и за потока на данни на самите профили на дадените типове потребители. По подобен начин се случват нещата и в DB Manager, където фигурира Professor DB Manager и Student DB Manager.

6. **Потребителите от счетоводния отдел, контролират финансовите операции, които засягат другите потребители (студентски такси, възнаграждения на служителите, и др.), както и разплащания с външни изпълнители на услуги.**

Реализирано е чрез Accounting Department в Server и неговите подмодули. И информацията относно плащанията се обработва в Payments DB Manager, преди да постъпи в базата.

7. **Студентите могат да се записват одобрени курсове, само в рамките на тяхната специалност и при условие, че профилът им отговаря на входните изисквания за компетентности за съответния курс.**

Курсовете и информацията свързана с тях преминава през Courses в Data Flow Module и Courses в Specialty manager. Компетентностите и входните изисквания се поемат отново от Security Manager.

8. **Студентите могат да генерират различни видове официални справки за студентския им статус: уверения, академични справки и т.н.**

Осъществено е чрез Reports Manager, който получава информацията за статуса от Data Flow Module (Student Profile).

9. **Официалните справки са електронни, като трябва да са защитени от опит за фалшифициране. При желание, справките може да се разпечатват и на хартия, като хартиеното копие трябва да има механизъм за верифициране с електронния вариант на справката.**

Защитата от опит за фалшифициране се извършва, чрез Security Manager, а верифицирането на електронния вариант се осъществява, чрез QR Code, който се генерира и на по-късен етап - разчита от подмодули на QR Code Manager.

10. **Системата да поддържа електронни студентски книжки, които са част от студентския профил. В тях, преподавателите внасят оценките на студентите по записаните от тях дисциплини, а студентите може да преглеждат своите книжки.**

Студентските книжки и операциите свързани с тях се извършват в Student Book, като ограничението на правата отново се поема от Security Manager.

11. **Системата да поддържа механизъм за публикуване на публични събития (еднократни курсове, състезания, събирания на групи по интереси и т.н.), които да може да се създават от всички потребители.**

Реализирано е чрез подмодул на Internal communication - Public event manager.

12. **Системата да поддържа възможност за обмяна на лични съобщения между потребителите.**

Осъществено е чрез Direct messages manager, като тъй като съобщенията са лични сме използвали тактика за да осигурим конфиденциалност на данните. Информацията от събщенията се приемат от Encryption в Security Manager.

13. **Потребителите от студентския съвет, както и преподавателите могат да създават заявки за различни искания, които се преглеждат и одобряват от потребителите в административния отдел.**

Заявките се поемат от модула Request Manager. Като съответно там се изпълняват необходимите операции.

14. **Системата да поддържа защита на всички лични и финансови данни от неоторизиран достъп.**

За да осигурим защита на лични и финансови данни, прилагаме тактика за сигурност тип устояване на атаки, а именно чрез автентикация и оторизация, които са осъществени, чрез подмодули на Security Manager.

15. **Системата да предоставя API (публичен интерфейс) за достъп до генерираните официални справки и публични събития.**

API е модул в External System Communicator. Чрез него подобряваме използваемостта, като API улеснява работата на чужди компании и дружества и техните системи с нашата собствена.

16. **Системата трябва да е достъпна 24/7, като изключение за поддръжка и планирано обновяване се допуска само по време на официални празници.**

За да се погрижим за отказустойчивостта на системата сме предприели две действия. Първо, използвали сме междинна тактика на активният излишък и на пасивният излишък, като целта е чрез Load-Balancing Server да разпределим равномерно информацията измежду повече сървъри. Самите сървъри не обработват една и съща информация, а работят паралелно. Това ще намали рискът от цялостен отказ на системата и също ще осигури качествени характеристики като надеждност, изправност и производителност на системата. Второ, използвали сме втора резервна база, чиято информация се обновява постоянно, едновременно с първата, като по този начин сме използвали именно тактика за осигуряване на наличност – активен излишък. При срив на първата база ще се задейства втората, като след това при възстановяването на първата, временно тя ще работи в режим на сянка.

17. **Системата трябва да прави връзка със следните външни системи:**

    a. **Държавни публични регистри за текущи студенти, към които периодично (напр. 2 пъти годишно) се изпраща информация за статуса на студентите. Изпращаната информация се контролира от потребителите от учебен и административен отдел.**
    b. **Система за контрол на национална агенция за приходите и данъчната администрация.**

c. **Система за управление на учебното съдържание (напр. Moodle, но може и да е друга система, която се употребява в конкретния университет).**
d. **Списъкът с външни системи, с които се прави връзка може да се увеличи в процеса на използване на системата.**

Комуникацията с външни системи се осъществява чрез модулите в External System Communicator, като сме използвали тактика с цел предотвратяване на ефекта на вълната. Имаме ограничаване на комуникацията и скриване на информацията, чрез отделните модули за комуникация с всяка външна система.

18. **Системата да може да издържа на пикови натоварвания (например увеличаване на потребителските заявки по време на кампания за записване на изборни дисциплини, вписване на оценки по време на сесия и т.н.), като може да обработва едновременно 1000 заявки в секунда.**

Устойчивостта на пикови натоварвания при множество потребителски заявки, е осъществена, чрез тактика за производителност. Приложена е чрез Load-balancing Server, който разпределя заявките между множество сървъри, за да избегне забавяне при потока на данни на бизнес логиката.

5. Допълнителна информация

5.1 Архитектурни драйвери

Имат за цел да ни насочат в изготвянето на архитектурата при вземане на важни решения. Също така ни помагат да установим какви тактики е възможно да приложим в случай на необходимост за да удовлетворим качествените изисквания.

1. Системата обслужва следните отдели в университета:
   a. Учебен отдел
   b. Счетоводен отдел
   c. Студентски съвет
   d. Административен отдел

Отделянето на съответните административни групи представящи университета е същинска част в изграждането на архитектурата на системата. Очевидно всеки отдел ще има собствени спецификации и ще изпълнява различни операции (функции) в зависимост от областта, в която е насочен. Затова спомага процеса по оформяне на архитектурата.

2. Всеки отдел предполага наличието на определен тип потребители. Освен това съществуват и администратори на системата, преподаватели и студенти.

В процеса за разработка на системата е важно да вземем предвид, че всеки потребител ще има отделна сесия. Като спрямо това, различните потребители ще имат характерни за своята сесия правомощия. Прави се с цел да се гарантира сигурността на данните на отделните потребители и да им се предостави достъп до характерните за тях ресурси. Точно затова е важно да се включи като важно архитектурно решение.

3. Системата поддържа профили на студентите и преподавателите, в които се записват техните данни, както и информация за техните компетентности.

Получената от профилите информация ще се съхранява в база от данни. Съответно важна част от изграждането на системата е да се поддържат профилите на отделните потребители. Данните в базата ще могат да се обновяват. Архитектурата трябва да е структурирана така, че да позволява безпроблемно използване на данните от базата.

4. Системата да поддържа механизъм за публикуване на публични събития (еднократни курсове, състезания, събирания на групи по интереси и т.н.), които да може да се създават от всички потребители.

Системата ще поддържа публикации. За тази цел е нужно да се добави съответен интерфейс, който да изпълнява тази функционалност. Това налага изменения в архитектурата.

5. Системата да поддържа възможност за обмяна на лични съобщения между потребителите.

Потребителите ще могат да пишат, редактират и пращат съобщения. Посредством съобщенията ще се осъществява връзка между отделните потребители. За тази цел ще се добави интерфейс за комуникация, който ще наложи изменения в архитектурата.

6. Системата да поддържа защита на всички лични и финансови данни от неоторизиран достъп.

При проектиране на архитектурата трябва да се вземе предвид, че данните на потребителите трябва да останат защитени. За да се подсигури това към системата ще се добави модул за сигурност.

7. Системата трябва да е достъпна 24/7, като изключение за поддръжка и планирано обновяване се допуска само по време на официални празници.

За да бъде системата постоянно налична е необходимо да приложим тактика за изправност. Затова ще добавим повече сървъри между, които ще се разпределят заявките. С това осигуряваме отказаустойчивост на системата. Така заявките на потребителите ще бъдат обработвани непрекъснато. Именно затова е необходимо да се включи при изготвяне на архитектурата.

8. Системата трябва да прави връзка със следните външни системи:

a. Държавни публични регистри за текущи студенти, към които периодично (напр. 2 пъти годишно) се изпраща информация за статуса на студентите. Изпращаната информация се контролира от потребителите от учебен и административен отдел.
b. Система за контрол на национална агенция за приходите и данъчната администрация.
c. Система за управление на учебното съдържание (напр. Moodle, но може и да е друга система, която се употребява в конкретния университет).
d. Списъкът с външни системи, с които се прави връзка може да се увеличи в процеса на използване на системата.

Съществено е системата да може да си комуникира с други външни системи. Важно е, при поява на нови актьори от страна на интегрираните системи, да продължи нормалният си процес на работа. С различните системи комуникацията ще е различна, затова е важно те да бъдат уточнени в процеса на изготвяне на архитектурата.

9. Системата да може да издържа на пикови натоварвания (например увеличаване на потребителските заявки по време на кампания за записване на изборни дисциплини,

вписване на оценки по време на сесия и т.н.), като може да обработва едновременно 1000 заявки в секунда.

Предвидено е, че системата е подготвена да се справи с пикови натоварвания.Това е пряко свързано с производителността на системата, като в този период очевидно ще има затруднения. Затова играе важна роля именно в изграждането на архитектурата. Необходимо е потока на данни да се разпредели равномерно върху множество сървъри, свързани помежду си, като ще използваме Load-balancing Server, който ще разпределя равномерно информацията измежду сървърите.