Курсова работа по Софтуерни архитектури и разработка на софтуер

UNIROLE

Документация на архитектурата на софтуерната система UNIROLE за управление на процесите и студентската информация в университет.

Изготвено от:

Теодора Иванова 62250

Благовест Конецовски 62262

Съдържание

1.Въведение

1.1. Обща информация за текущия документ

1.1.1. Предназначение

Този документ представлява описание на архитектурата на софтуерната система за управление на процесите и студентската информация в университет. Основната му задача е да предостави в детайли и по достъпен за всички заинтересовани лица начин решенията, които са взети по отношение на проектирането на настоящата архитектура.

1.1.2. Използвани структури за изграждане на архитектурата

1. **<u>Декомпозиция на модулите</u>**- фундаментална за архитектурата структура, тъй като показва от статична гледна точка основните модули, от които е изградена системата, които са всъщност носителите на всяка една функционалност, която тя притежава. Изборът за представяне на тази структура е повлиян от факта, че тя носи едно концептуално виждане за това- какво представлява системата отвътре, като чрез декомпозирането се достига до най-ниско ниво на абстракция, т.е. до най-просто устроените модули. ***UNIROLE*** най-общо казано се разделя на три базови модула:
    - *User Interface*- съдържа модули, отговорни за потребителския интерфейс на различните видове потребители, използващи системата
    - *Business Logic*- съдържа модули, в които е реализирана логиката за различните функционалности, които системата предоставя на своите потребители
    - *Data Base Configuration Files*- съдържа модули, представляващи конфигурационни файлове, които

имат отговорността както да извличат информация от базата, така и да й предоставят такава.

Тези модули са взаимносвързани помежду си, като най-просто казано, осъществяват се двупосочни връзки- потребителите осъществяват желаните от тях операции чрез използването на възможно най-удобния потребителски интерфейс, техните заявки се обработват от системата, благодарение на бизнес логиката й, а данните за самите процеси, протичащи в системата, се съхраняват в локална база данни, която се достъпва чрез конфигурационни файлове.

2. **Структура на внедряването**- Структурата на внедряването е ключова за архитектурата на системата ***UNIROLE***, защото показва по какъв начин отделните модули на системата се разполагат върху хардуерните елемнти. Наблюдава се наличието на редица сървъри, върху които се помещават основните функционалности на системата. Изборът за представяне на тази структура е повлиян от факта, че тя носи едно концептуално виждане за това- как се удовлетворяват някои от качествените изисквания към системата чрез избора на подходящите хардуерни машини.
3. **Структура на процесите**- В настоящата документация се наблюдават 2 структури на процесите, които показват по-важните и представляващи интерес за заинтересованите лица процеси. Изборът за представяне на този вид структура е повлиян от факта, че ***UNIROLE*** като система за управление на процесите в университет предполага приемането на множество потребителски заявки и навременното им изпълняване, което само по себе си обуславя наличието на множество процеси, протичащи

вътре в системата. Двете структури представят следните процеси:

- Структура на процесите- Записване за избираема дисциплина от студент
- Структура на процесите- Вписване на оценка от преподавател на студент в студентската му книжка

1.1.3. Структура на документа

- Въведение- заема секция 1 от текущия документ и представлява неговата уводна част. В нея са описани общата информация за документацията (секция 1.1), най-общи сведения за системата, която е документирана чрез настоящото описание (секция 1.2) и терминологичният речник, даващ разяснения относно изоплзваните термини (секция 1.3).
- Декомпозиция на модулите- заема секция 2 от текущия документ, в която е представена фундаментална за архитектурата структура, тъй като тя показва от статична гледна точка основните модули, от които е изградена системата, които са всъщност носителите на всяка една функционалност, която системата притежава. Тази структура е разгледана както от най-високо ниво на абстракция- т.е. първичният изглед (секция 2.1), без излишни детайли, така и всеки един от модулите бива обстойно изследван до най-ниско ниво (секция 2.3). В тази секция може да бъде видяна и т.нар. контекстна диаграма, която дава възможност системата да бъде видяна от гледна точка на външните системи, с които си взаимодейства (секция 2.2).
- Описание на допълнителните структури- заема секция 3 от настоящия документ, в която са

представени избраните допълнителни структури (в случая на настоящия документ това са структура на внедряването и две структури на процесите). Всяка структура е разгледана както от най-високо ниво на абстракция- т.е. първичният изглед (секция 3.1), без излишни детайли, така и всеки един от модулите и връзките биват обстойно изследвани до най-ниско ниво (секция 3.2). Предлага се също така описание на обкръжението- т.е. всички онези системи (ако има такива), с които даден модул взаимодейства(секция 3.3).

- Архитектурна обосновка- ключова за документацията на архитектурата секция, тъй като дава подробно обяснение на това- по какъв начин взетите архитектурни решения повлияват върху удовлетворяването на всички изисквания към системата.
- Допълнителна информация- съдържа всичко онова, което би могло да се съотнесе към разработката на софтуерната архитектура на системата ***UNIROLE,*** но не намира място в гореописаните секции.

1.2. Общи сведения за системата

Системата ***UNIROLE*** представлява система за управление на процесите и студентската информация в университет. Предназначена е както за студенти и преподаватели, така и за служители от административни и счетоводни отдели на съответния университет. Възможностите, които системата предоставя на различните видове потребители са:

- Студенти- информация за студентския статус, записване на курсове, преглед на оценки, преглед и записване за различни събития, осъществяване на контакти със студенти и преподаватели и други

- Преподаватели- информация за статуса, информация за техните студенти, вписване на оценки на студент, осъществяване на контакти със студенти и преподаватели и други
- Служители от административен отдел- чрез системата те извършват нужния контрол на всички дейности в университета
- Служители от счетодовдния отдел- чрез системата те извършват всички финансови операции, случващи се в университета

Някои от качествените изисквания към системата са:

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

1.3. Терминологичен речник

1.3.1. Специфични термини

- **База от данни** – колекция от информация, организирана така, че да може лесно да се достъпва, упрявлява и организира.
- **Сървър** – стартирана инстанция на софтуерна система, която може да приема заявки от клиент и да връща подходящи отговори
- **Приложение** - софтуер, предназначен да помогне на потребителя да извърши определена задача

- **Интерфейс** - споделена граница, между която два отделни компонента на компютърна система си обменят информация
- **Модул** – логически обособена софтуерна единица
- **Процес** - съвкупност от стъпки, която изгражда логическо действие и стига определена цел
- **Декомпозиция на модули** – софтуерна структура, показваща разделянето на системата на подмодули, като разделянето продължава, докато не се стигне до максимално опростени и лесни за разбиране единици.
- **Структура на процесите** - софтуерна структура, показваща какви са етапите, през които преминава, и условията, на които трябва да отговаря, даден процес.

1.3.2. Списък на софтуерните елемнти

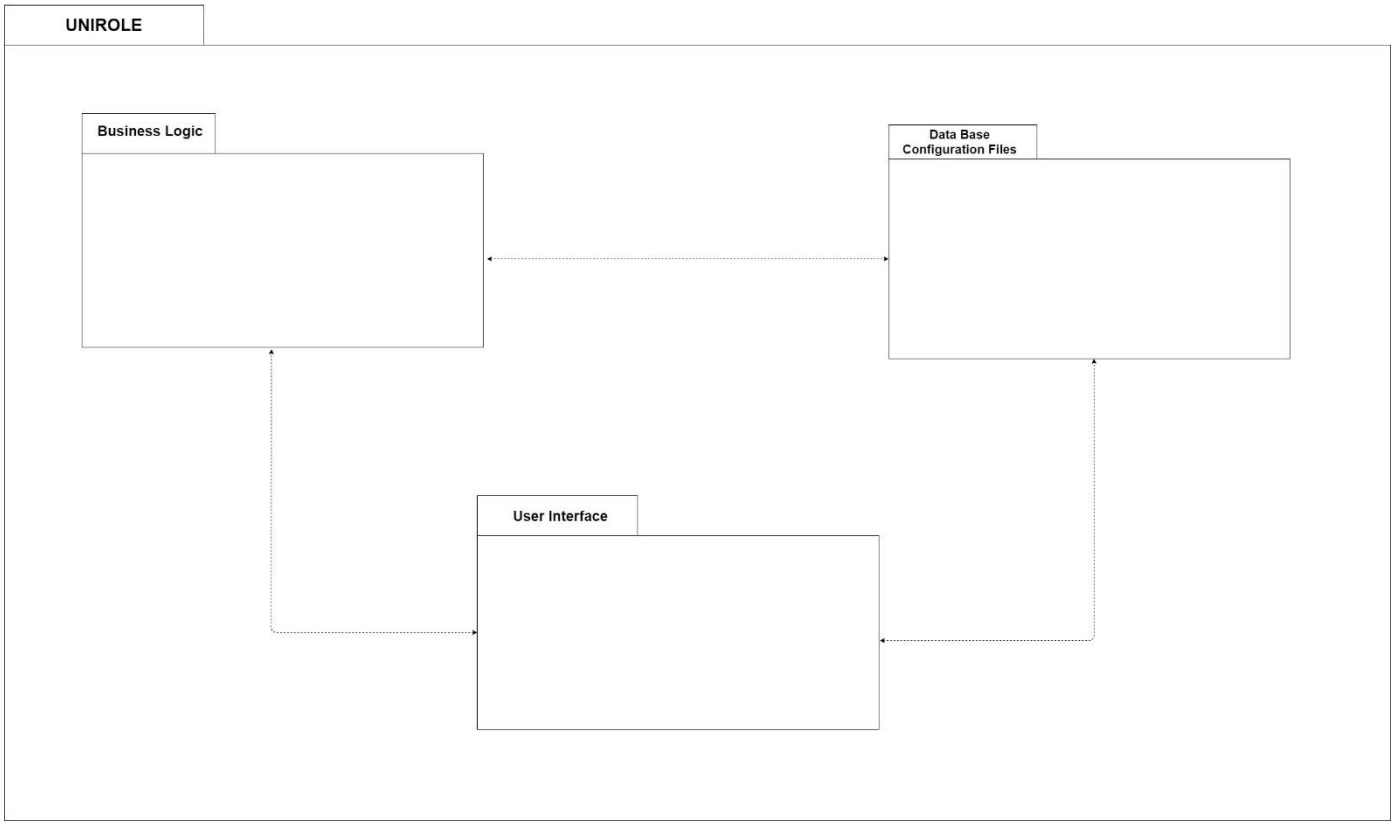
- Business Logic
    - User Manager
        - Log In
        - Event Creation
        - Message Exchange
        - Student Book Insertion
        - Request Creation
        - Request Acception
        - Official Reference Creation
        - Student Book View
    - Mobile App
    - Web App
    - Controller
    - Payments
        - Card
        - Paypal
    - Account Generator
    - Authentication

- Outer Systems Connector
- User Interface
  - System Administrator UI
  - Students UI
    - Student Concil UI
  - Students Book UI
  - Teachers UI
  - Administration UI
    - Academic Department UI
    - Accountants UI
  - Visitors UI
- Data Base Configuration Files
  - Users
    - Administration Information
      - Academic Department Information
      - Accountants Information
    - System Administration Information
    - Messages History
    - Students Information
    - Teachers Information
  - Finance
    - Salaries
    - Students Fees
    - Other Payments
  - Security
  - Events
  - Courses
  - Official References Information
  - Student Book Content
    - Insertion
    - View
  - Request History
  - Specialities

- Distributor

2. Декомпозиция на модулите

2.1. Общ вид на декомпозицията на модулите на системата



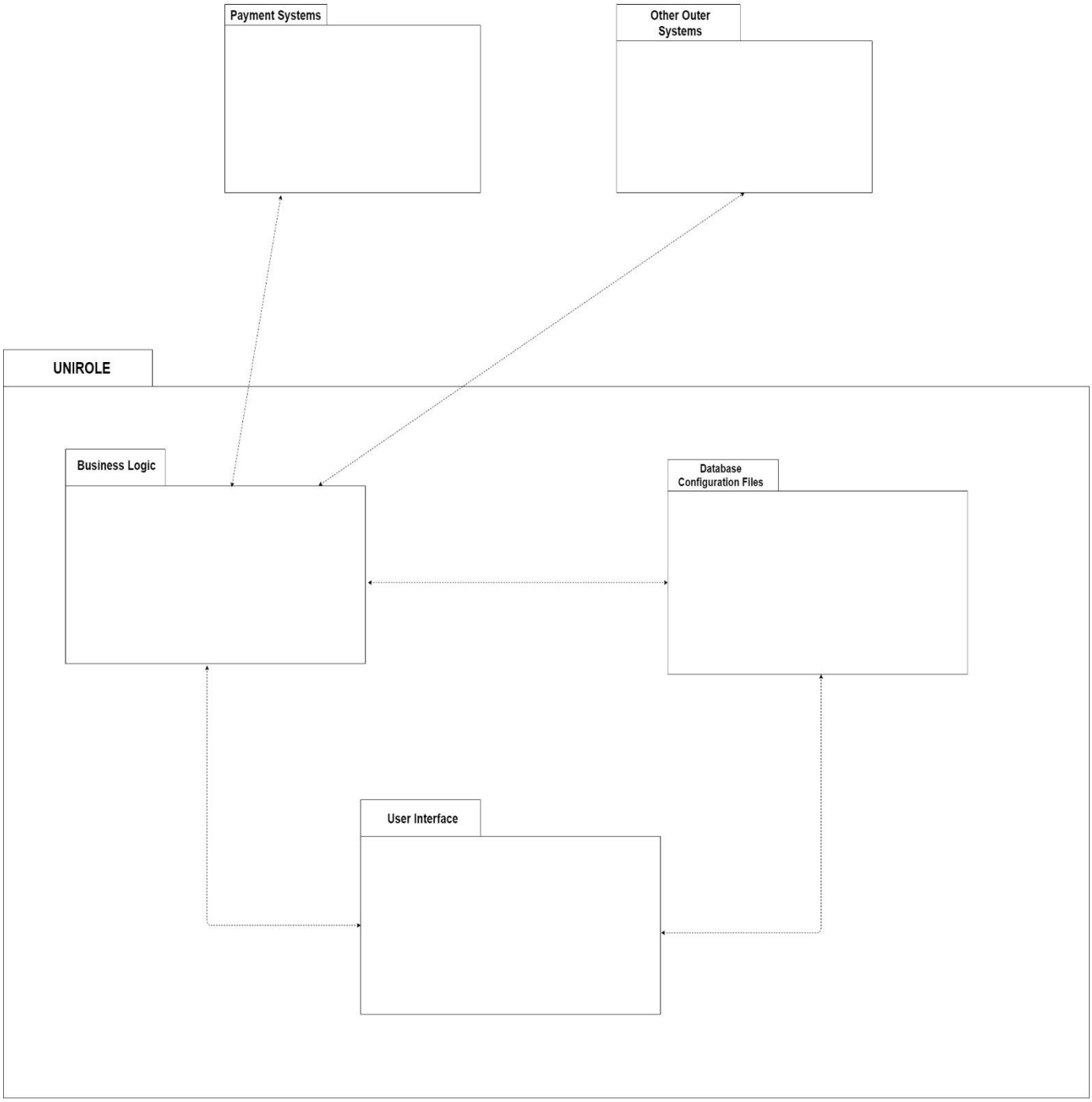
Тази структура е фундаментална за архитектурата, тъй като показва от статична гледна точка основните модули, от които е изградена системата, които са всъщност носителите на всяка една функционалност, която тя притежава. Изборът за представяне на тази структура е повлиян от факта, че тя носи едно концептуално виждане за това- какво представлява системата отвътре, като чрез декомпозирането се достига до най-ниско ниво на абстракция, т.е. до най-просто устроените модули.
UNIROLE най-общо казано се разделя на три базови модула:

- *User Interface*- съдържа модули, отговорни за потребителския интерфейс на различните видове потребители, използващи системата

- *Business Logic*- съдържа модули, в които е реализирана логиката за различните функционалности, които системата предоставя на своите потребители
- *Data Base Configuration Files*- съдържа модули, представляващи конфигурационни файлове, които имат отговорността както да извличат информация от базата, така и да й предоставят такава.

Тези модули са взаимносвързани помежду си, като най-просто казано, осъществяват се двупосочни връзки- потребителите осъществяват желаните от тях операции чрез използването на възможно най-удобния потребителски интерфейс, техните заявки се обработват от системата, благодарение на бизнес логиката й, а данните за самите процеси, протичащи в системата, се съхраняват в локална база данни, която се достъпва чрез конфигурационни файлове.

2.2. Контекстна диаграма

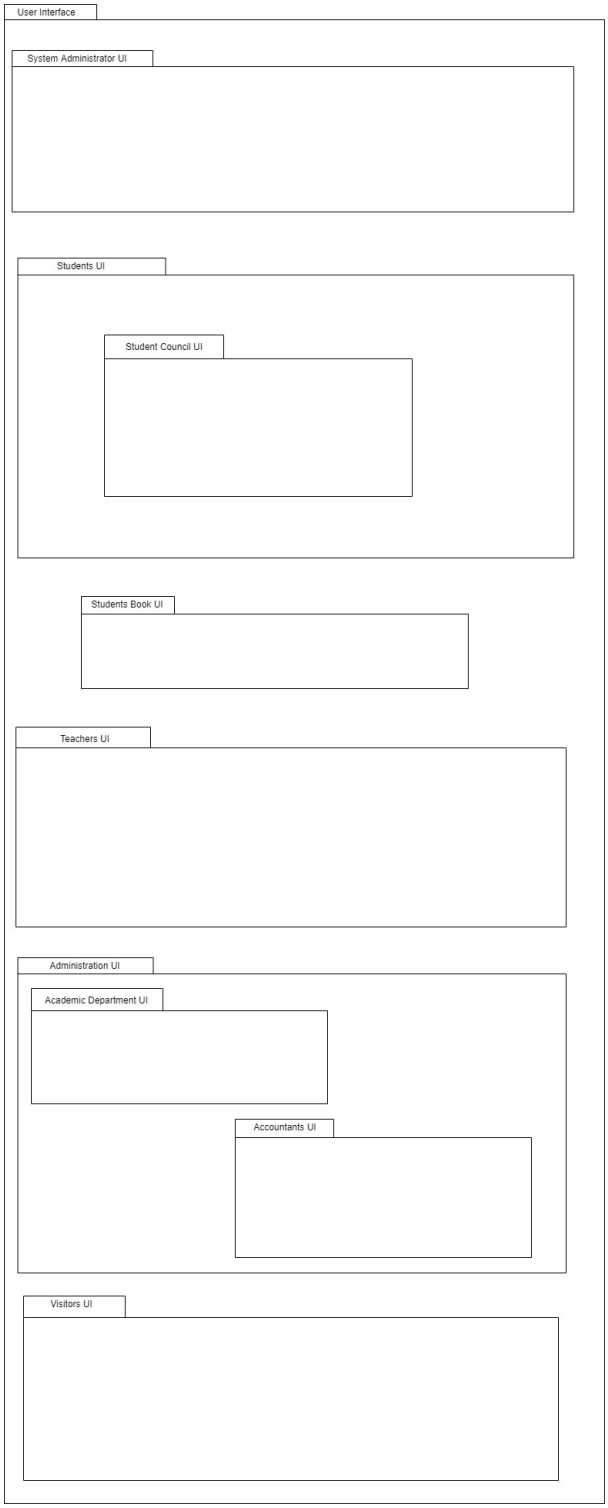


Описание на обкръжението

- Модулът **Database Configurartion Files** обединява набор от конфигурационни
  файлове, в които се съхраняват заявките към базата за вкарване на информация в нея и за извличане на такава от нея. Основната идея е, че базата данни, намираща се на локална мрежа, не се достъпва директно с цел защита от атаки.
- Модулът **Business Logic** комуникира с външни системи за извършване на разплащанията в системата, а също и с други външни системи, до които системата се обръща при необходимост от допълнителна информация за студентите или преподавателите.

2.3. Подробно описание на всеки модул

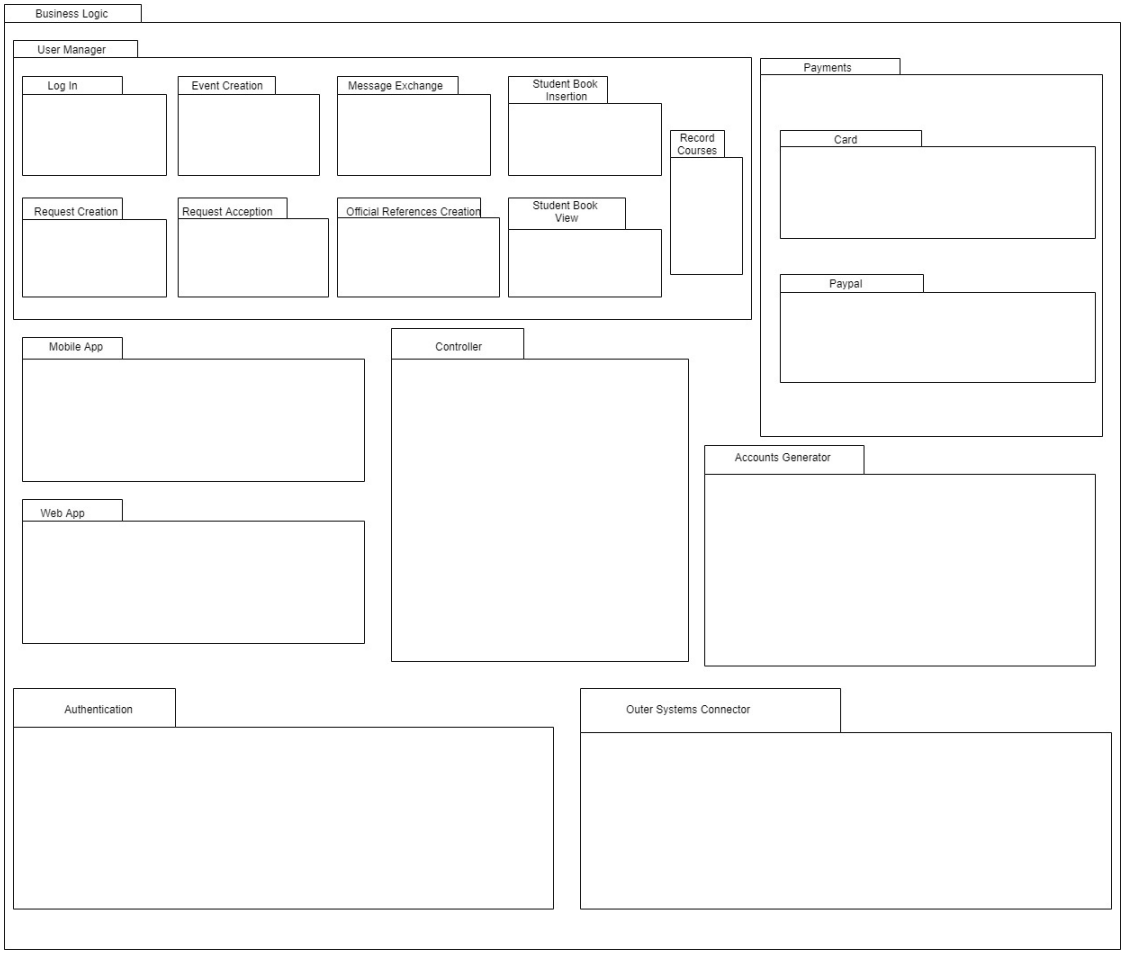
User Interface



UI 2.3.1. Предназначение на модула
Системата предоставя различни интерфейси в зависимост от типа потребител, влязъл в нея. Това се дължи на факта, че използващите системата потребители са разнообразни и поради тази причина всеки от тях получава различни функционалности, съобразени с конкретния тип потребител, който се определя от правата и задълженията на съответния човек в университета.

UI 2.3.2. Основни отговорности на модула
Да предостави достъпна визуална част на различните потребители, благодарение на която те боравят успешно с функционалностите на системата. Този модул комуникира с Controller-а, за да се осъществи необходимата операция, заявена от потребителя.

Business Logic



BL 2.3.1. Предназначение на модула
Това е модул, който съдържа цялата логика на системата. Всички функционалности, които системата съдържа, са описани чрез отделните подмодули на този модул.

BL 2.3.2. Основни отговорности на модула
Да предостави възможността на системата да изпълни своята функционалност.

BL 2.3.3. Описание на интерфейсите на модулите

User Manager

Log In

- public void login(String username, String password); - вход в системата
  - входни данни: потребителско име и парола
  - резултат от изпълнението: получаване на достъп до системата с правата на съответния потребител
  - грешки и изключения: при невалидни данни се изпраща подходящо съобщение и потребителят не получава достъп до системата
  - зависимости от други елемнти: след въвеждане на входните данни се извършва идентификация на потребителя (връзка с модула **Authentication**), която, от своя страна, е възможна чрез обръщането към базата данни за проверка дали за дадения потребител съществува запис в нея.

Record Courses

- public List<Course> findCoursesUserSpeciality(Student student);
  - входни данни: обект от тип Student
  - изходни данни: списък от позволени за дадената специалност и курс дисциплини
  - грешки и изключения: при невалидни данни се изпраща подходящо съобщение

- private void showCoursesUserSpeciality(List<Course> courses);
    - входни данни: списъкът, върнат от findCoursesUserSpeciality(Student student);
    - резултат от изпълнението: извежда намерените курсове
    - грешки и изключения: извежда съобщение при празен списък от специалности
- public Course chooseCourse(Student student, List<Course> courses);
    - входни данни: обект от тип Student, списък от избрани курсове
    - изходни данни: обект от тип Course (избраният от студента курс)
- public boolean checkNumberEnrolledCourses(Student student);- проверка за броя избрани курсове
    - входни данни: обект от тип Student
    - изходни данни: true- ако избраните курсове са < 5, false- иначе
- public boolean checkRequirements(Course choosenCourse, Student student);
    - входни данни: избраният курс и обект от тип Student
    - изходни данни: true- ако критериите са изпълнени, false- иначе

<u>View Students Book</u>

- public void viewStudentBook(); - отваряне на студентска книжка за преглед (функционалност, достъпна на потреителите преподаватели и на потребителите студенти)
    - входни данни: няма
    - изходни данни: отваряне на нов интерфейс, представляващ студентската книжка на дадения потребител
    - грешки и изключения: такива може да се появят евентуално, ако има някакъв срив в системата,

или се осъществява някакъв вид техническа проверка
    - зависимости от други елемнти: при заявка за преглед на студентската книжка се осъществява връзка със **Students Book Content**

Students Book Insertion

- public void insertInStudentsBook(int fn, double grade); - отваряне на студентска книжка за въвеждане на оценка (функционалност, достъпна на потребителите преподаватели)
    - входни данни: факултетен номер на студент и оценка за вписване
    - изходни данни: отваряне на нов интерфейс, представляващ студентската книжка на дадения студент, чийто факултетен номер е въведен
    - грешки и изключения: при въведена некоректна оценка потребителят е уведомен със съобщение
    - зависимости от други елемнти: при заявка за преглед на студентската книжка се осъществява връзка със **Students Book Content**

Request Creation

- public void createRequest(String message); - отваряне на поле за въвеждане на желаното искане
    - входни данни: съобщението, което да бъде изпратено
    - изходни данни: няма (те се очакват от модула **Request Acception**)
    - грешки и изключения: при некоректно съдържание на съобщението потребителят е уведомен чрез съобщение за грешка

- зависимост от други елементи: информацията за направените заявки се пази в базата в модула **Requests History;** одобряването или отхвърлянето на заявките става чрез логиката в модула **Request Acception**

Request Acception

- public boolean acceptRequest(Request request);
  - входни данни: заявката, която трябва да бъде одобрена или отхвърлена
  - изходни данни: true, ако заявката е одобрена, false- в противен случай
  - грешки и изключения: ако подадената за обработка заявка е невалидна потребителят е уведомен със съобщение за това
  - зависимост от други елементи: заявките се получават благодарение на съхранената в базата данни информация в модула **Requests History**

Event Creation

- public bool createEvent(String name, String description, Date date, double duration, double fee); - създаване на публично събитие
  - входни данни: име на събитието, описание на събитието, дата и час на провеждане, продължителност на събитието и такса за участие в него
  - изходни данни: true при успешно създадено събитие; false- в противен случай
  - грешки и изключения: при некоректно съдържание на някоя от входните данни потребителят е уведомен чрез съобщение за грешка

        - зависимост от други елемнти: информацията за генерираните събития се пази в базата в модула **Events**
    - public void publishEvent(Event event);- публикуване на вече създадено събитие
        - входни данни: информация за събитието, която се взима от информацията, запазена в конфигурационните файлове
        - резултат от изпълнението: създаденото събитие е публикувано и видимо за всички потребители
        - зависимост от други елемнти: информацията за генерираните събития черпи от базата от модула **Events**
    - public void editEvent(Event event); - редактиране на публикувано събитие
        - входни данни: информация за събитието, която се взима от информацията, запазена в конфигурационните файлове
        - резултат от изпълнението: дава се възможност за коригиране на публикувано събитие
        - зависимост от други елемнти: информацията за генерираните събития се взима от модула **Events** и съответно се обновява и там

Message Exchange

- public void sendMessage(User anotherUser);- изпращане на съобщения между потребителите
    - входни данни: потребител, до когото да бъде изпратено съобщението
    - резултат от изпълнението: изпращане на съобщението до съответния потребител
    - грешки и изкючения: при евентуална невъзможност съобщението да бъде изпратено потребителят бива известен със съобщение за грешка

        - зависимост от други елемнти: история на съобщенията между различните потребители се пази в базата данни и се достъпва благодарение на логиката в модула **Message History**
    - public void attachDocument(File filename, User anotherUser);- прикачване на документ (текстов, снимка и т.н.) към съобщение
        - входни данни: прикаченият файл и потребител, на когото да бъде изпратен
        - резултат от изпълнението: прикачване към чата и възможност за изпращане
        - грешки и изкючения: при евентуална невъзможност съобщението да бъде изпратено потребителят бива известен със съобщение за грешка
    - public void removeAttachedDocument(File filename, User anotherUser);- премахване на документ (текстов, снимка и т.н.), прикачен към съобщение, преди да бъде изпратено
        - входни данни: прикаченият файл и потребител, на когото да бъде изпратен
        - резултат от изпълнението: премахване на прикачен файл
        - грешки и изкючения: при евентуална невъзможност файлът да бъде прикачен потребителят бива известен със съобщение за грешка
    - public void removeMessage(User anotherUser, int messageID);- взима съобщение, което е вече изпратено до anotherUser и съответно е запазено в конфигурационните файлове в модула **Message History**, и го премахва посредством messageID
        - входни данни: потребителят, до когото е изратено съобщението, и ID-то на даденото съобщение

    - резултат от изпълнението- премахване на съобщението от чата и от базата съответно (само ако резултатът от изпълнението на долната функция е true)
    - грешки и изкючения: при евентуална невъзможност съобщението да бъде изтрито (резултатът от изпълнението на долната функция е false) потребителят бива известен със съобщение за грешка
    - зависимост от външни елементи: комуникира с модула **Message History**, използва функцията checkIfRemovable(User anotherUser, int messageID)
- public boolean checkIfRemovable(User anotherUser, int messageID); - проверява дали съобщението е било изпратено преди по-малко от 5 минути. Ако да- връща true и съобщението може да бъде премахнато, иначе- връща false и операцията е неуспешна.
    - входни данни: потребителят, до когото е изратено съобщението, и ID-то на даденото съобщение
    - изходни данни: true- ако съобщението може да бъде премахнато, false- иначе
- public void editMessage(User anotherUser, int messageID);- взима съобщение, което е вече изпратено до anotherUser и съответно е запазено в конфигурационните файлове в модула **Message History**, и позволява редактиране, като намира даденото съобщение в базата посредством messageID
    - входни данни: потребителят, до когото е изпратено съобщението, и ID-то на даденото съобщение
    - резултат от изпълнението- премахване на съобщението от чата и от базата съответно (само ако резултатът от изпълнението на долната функция е true)

        - грешки и изкючения: при евентуална невъзможност съобщението да бъде изтрито (резултатът от изпълнението на долната функция е false) потребителят бива известен със съобщение за грешка
        - зависимост от външни елементи: комуникира с модула **Message History**, използва функцията checkIfEditable(User anotherUser, int messageID)
    - public boolean checkIfEditable(User anotherUser, int messageID); - проверява дали съобщението е било изпратено преди по-малко от 5 минути. Ако да- връща true и съобщението може да бъде редактирано, иначе- връща false и операцията е неуспешна.
        - входни данни: потребителят, до когото е изратено съобщението, и ID-то на даденото съобщение
        - изходни данни: true- ако съобщението може да бъде премахнато, false- иначе

<u>Official References Creation</u>

- public void createOfficialRef(int FN); - генериране на официална справка за студентския статус на даден студент
    - входни данни: факултетен номер на студент, заявил справка за статуса си
    - изходни данни: генериране на официална справка
    - грешки и изключения: при неправилен факултетен номер или при невъзможност да бъде генерирана справка поради някакъв проблем потребителят е уведомен със съобщение за грешка
    - зависимост от други елемнти: информацията за генерираните събития се пази в базата в модула **Official References Information**

Payments

- public boolean paySemesterFeeByCard(double money, String IBAN, String cardOwner, String backNums);- плащане на семестриална такса чрез карта
  - входни данни: сума, която да се преведе; IBAN на картата; притежател на картата; числата на гърба на картата
  - изходни данни: true, ако преводът е извършен успешно, false, в противен случай
  - грешки и изключения: при некоректност на някоя от входните данни или при невъзможност да бъде извършено плащането се извежда съобщение за грешка
  - зависимост от други елементи: комуникация с външни системи за разплащане; отразяване в базата посредством модула **Salaries**
- public boolean paySemesterFeeByPayPal(double money, String IBAN, String cardOwner, String backNums);- плащане на семестриална такса чрез PayPal
  - входни данни: сума, която да се преведе; IBAN на картата; притежател на картата; числата на гърба на картата
  - изходни данни: true, ако преводът е извършен успешно, false, в противен случай
  - грешки и изключения: при некоректност на някоя от входните данни или при невъзможност да бъде извършено прлащането се извежда съобщение за грешка
  - зависимост от други елементи: комуникация с външни системи за разплащане; отразяване в базата посредством модула **Salaries**

- public boolean giveSalary(String EGN);- превод на заплата на преподавател
  - входни данни: ЕГН на преподавател
  - изходни данни: true, ако преводът е извършен успешно, false, в противен случай

        - грешки и изключения: при невъзможност да бъде извършен преводът се извежда съобщение за грешка
        - зависимост от други елементи: комуникация с външни системи за разплащане; отразяване в базата посредством модула **Salaries**
    - public boolean giveScholarship(int FN);- превод на стипендия на студент
        - входни данни: факултетен номер на студент
        - изходни данни: true, ако преводът е извършен успешно, false, в противен случай
        - грешки и изключения: при некоректност на факултетният номер или при невъзможност да бъде извършен преводът се извежда съобщение за грешка
        - зависимост от други елементи: отразяване в базата посредством модула **Students Fees**

Mobile App- модул, в който се съдържа логиката за връзка на системата с мобилното приложение.

Web App- модул, в който се съдържа логиката за връзка на системата с приложението, отворено през Web браузър.

Controller- модул с изключителна роля за системата, тъй като той е отговорен за приемане на подадените заявки от потребителите посредством User Interface-а и препращането им към правилния модул.

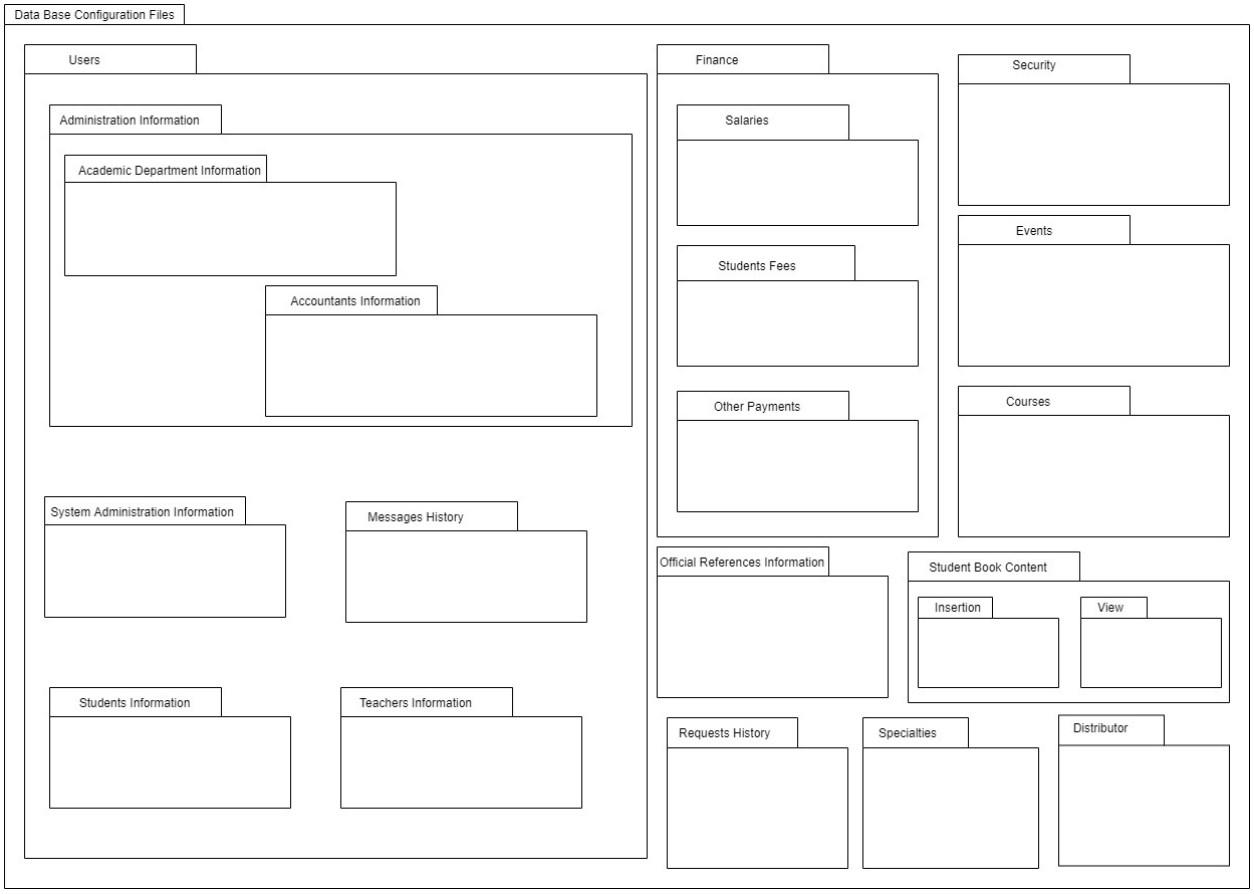
Authentication- модул, който е отговорен за верифициране на самоличността на потребителите, когато използват опцията за вход в системата (логиката в Log In модула). Изборът да бъде отделен подмодул на модула Business Logic е повлиян от съображението, че при евентуална атака от типа: Множество заявки за вход, дори Authentication модула да се срине, всички потребители, които вече са в системата, няма да бъдат повлияни от случилата се атака.

Accounts Generator

- public void createAccounts(Hashmap<String, String> userInfo);- при записване на новоприети студенти се генерират техните акаунти посредством имената и ЕГН-тата им
  - входни данни: хешмап от имената и ЕГН-тата на съответния студент
  - изходни данни: създаденият акаунт
  - грешки и изключения: при невъзможност да бъде създаден акаунт се уведомяват администраторите на системата
  - зависимост от други елементи: отразяване в базата посредством модула **Students Information**

Outer Systems Connector- - модул, служещ за връзка на системата със съответните външни системи.

Database Configuration Files



DB 2.3.1. Предназначение на модула

В този модул се съдържат подмодули, които представляват отделни файлове, благодарение на които базата данни на системата се достъпва и се извилча необходимата информация от нея.

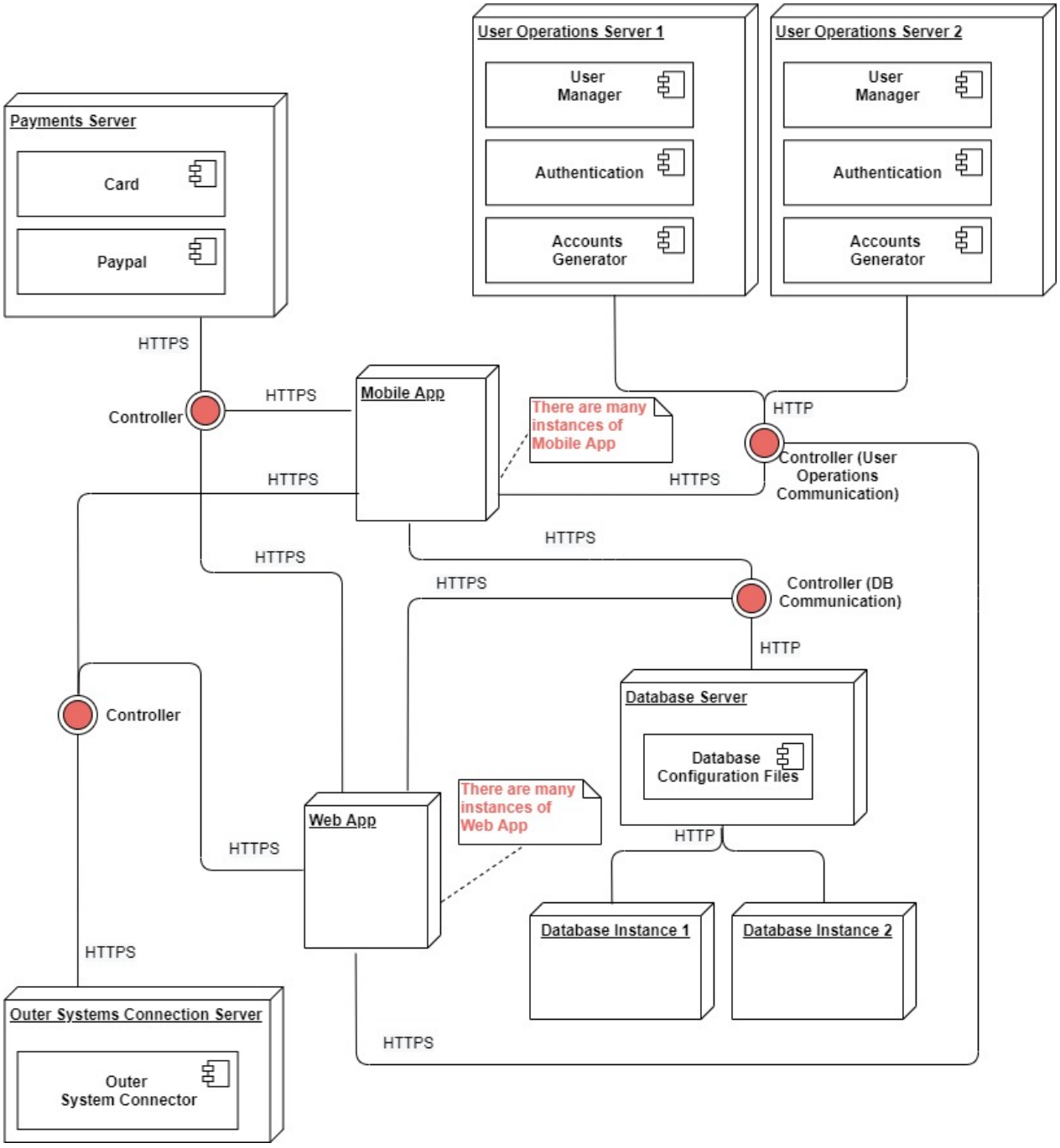
DB 2.3.2. Основни отговорности на модула

Всички направени заявки за достъп до базата и извличане на информация от нея логично са последвани от връщане на отговр от базата, който се разпраща до съответните модули, подали заявките.

3.Описание на допълнителните структури

D3.Структура на внедряването

D3.1. Първично представяне



Структурата на внедряването е ключова за архитектурата на системата UNIROLE, защото показва по какъв начин отделните модули на системата се разполагат върху хардуерните елемнти.
Наблюдава се наличието на редица сървъри, върху които се помещават основните функционалности на системата.
Също така някои по-важни подмодули, които са част от един от основните модули (Business Logic): User Manager (заедно с операциите на Authentication и Accounts Generator), Outer System Connector, Payments, са разположени на отделни сървъри с цел устояване на високата натовареност на системата и удовлетворяване на качественото изискване системата да е налична 24/7.
От структурата се вижда, че са налични две инстанции на базата данни на системата- това са копия на информацията, която се съдържа в базата. Те са две, за да се разпредели равномерно натоварването от множеството заявки, отправени към базата, като това ще спомогне и за паралелното обработване на заявки. Заявките се получават и разпределят от Database Server, който получава заявките от останалите модули и ги изпраща към подходящата инстанция на базата.
Изборът структурата на внедряването да изглежда по този начин, е повлиян от някои от качествените изисквания, на които системата трябва да отговаря. Едно такова изискване е:
Системата да поддържа защита на всички лични и финансови данни от неоторизиран достъп.
То обуславя решението копието на информацията от базата да е на отделни сървъри, а също и всички финансови данни биват защитени чрез обособяването на отделен сървър за плащанията (Payments Server).
Тъй като системата трябва да кореспондира с множество външни системи, добра идея е логиката за тази кореспонденция да бъде на отделен сървър. Така при евентуално претоварване или опит за атака да не се срине цялата система, а само този сървър.
Едно от качествените изисквания, което засяга главно потребителите на системата, е:
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.

То е причината операциите, свързани с обработка на заявките на потребителите, чиято логика се съдържа в User Manager модула, да са отделени на двата сървъра User Operations Server 1/2, за да издържа системата на натоварването в тези периоди.
HTTP и HTTPS са използваните протоколи за обмен на данни. Както е известно, HTTPS е Secured протокол, по който данните преминават в криптиран вид. Затова и по него преминават заявките от потребителите (Mobile App и Web App), докато достигнат до Controller-а. Вътре в системата данните са необходими в суров вид (некриптирани) и затова се използва HTTP протокол.
Протокли HTTPS се наблюдават от конролерите до Payments и Outer System Connector сървърите съответно, защото е необходима защита на данните.

D3.2.Описание на елементите и връзките

User Operations 1/2 – Тези сървъри се занимават с обработката на цялостната логистика на User Manager, като Controller- модул намиращ се на User Manager Server разпределя равномерно заявките към двата сървъра- User Operations 1 и User Operations 2, за да се разпредели натоварването при многобройни заявки по време на ключови събития като:
-записване за избираеми дисциплини (натоварването се получава от множеството студенти)
-нанасяне на оценки(натоварването се получава от множеството преподаватели, нанасящи оценки на всички свой студенти)
-генериране на нови акаунти(натоварването се получава поради големия брои новоприети студенти, на които трябва да се предоставят акаунти за достъп в системата)
Payments Server – Съдържа модула за плащане. Отделен е на отделна машина с цел повишаване на сигурността. При евентуална атака над другите сървъри да не може да се достигне непосредствено да Payments Server
Outer Systems Connection Server – Съдържа модула за връзка с външни системи. Отделен е на отделен сървър , за да се осигури по-висока сигурност за запазване на данните, който се обменят между системата и другите външни системи.
Database Server – Сървър, който е разпределител на множеството заявки към Database Instance 1 и Database Instance 2, като осигурява равномерно

натоварване, с цел обработка на повече заявки за единица време. Другите модури комуникират непосредствено с него, за да получат нужната им информация от базата данни.
Database Instance 1/2 – сървари с копие на базата данни. Има две инстанции, за да може да се отговаря на по-голямо натоварване. Заявките се получават от Database Server.Намират се на локална мрежа, за да не могат да се достъпят от външния свят.Самата база данни се намира на друга машина, с цел да няма достъп до нея никой ог външния свят, а само Database Instance 1/2 , които са на същата локална мрежа. Структурата е реализирана по този начин, за да се пази главтана база данни от евентуална атака с цел изтриване на данни.
Web App – Това са всички отделни потребители, влезли през Web browser. Те могат да бъдат на отдалечени устройства, без значиние от операционната им система, както и браузъра, който използват, за да могат да използват всички функционалностти на системата.
Mobile App – Това са всички отделни потребители, влезли през мобилни устройста,без значиние от операционната им система(iOS,Android и тн), за да могат да използват всички функционалностти на системата.
Controller модулът служи за разпределение на заявките от потребителите към самата система. Логиката в модула Controller е разпределна на три сървъра:

1. На един сървър се намира контролерът, който отговаря за разпределението на заявките към Outer System Connector Server и Payments Server
2. На втори сървър се намира контролерът, който отговаря за разпределенето на заявките към User Operations сървърите
3. На трети сървър се намира контролерът, който отговаря за разпределенето на заявките към Database Server

Целта на разделянето на логиката на контролера е да се разпредели равномерно натоварването му, да се предотврати срив на системата, породен от множество заявки и да се осигури непрекъснатата наличност на системата.

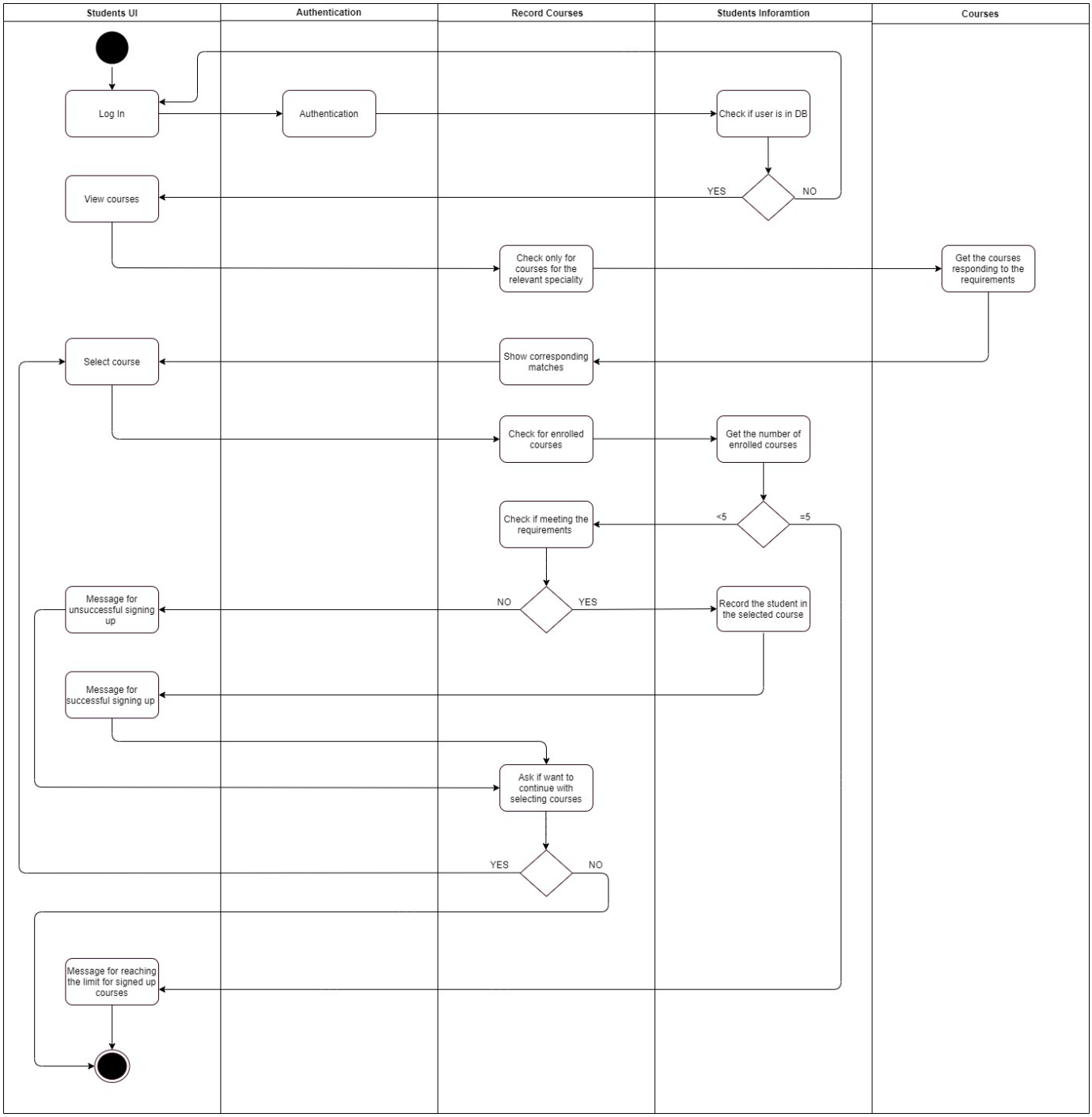
D3.3. Описание на обкръжението

Payments Server – комуникира с множество външни системи, за да се осъществят необходимите транзакции.

Outer System Connector – комуникира с множество външни системи, за да взема необходимата информация и да предава такава.

FP3.Структура на процесите: Записване на курс от студент

FP3.1. Първично представяне



Процесът по записване на изибраема дисциплина от студент е визуализиран благодарение на горната диаграма на активностите. Чрез нея се показва как е постигнато едно от функционалните изисквания към системата, а именно:

> *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.*

По-общо, заключенията, които биха могли да се направят чрез недотам задълбочен анализ, са следните:

1) Потребителският интерфейс е разбираем и интуитивен, тъй като потребителите лесно могат да открият необходимите функционалности, за да извършат исканите от тях операции. В допълнение, системата е ангажирана и с дейността по уведомяване на потребителите при всякакви проблеми, възникнали по време на изпълнение, с което ги държи информирани.
2) Бизнес логиката е свързващото звено между това, което потребителите виждат, и конфигурационните файлове за връзка с базата от данни. В нея се извършват всевъзможни проверки с цел покриване на всички случаи.
3) В конфигурационните файлове, служещи за връзка с базата, се извършват подходящите за текущия процес заявки към базата.

FP3.2.Описание на елементите и връзките

При желание на студент да запише избираеми дисциплини той, първо, трябва да е влязъл в профила си. При вход в системата се извършва удостоверяване на самоличността на потребителя, което изисква проверка за наличността на данни за него в базата. Ако не присъства в базата данни, се приканва да въведе валидни потребителско име и парола. Иначе получава възможността да прегледа курсовете, налични за неговата специалност. За да се осъществи коректния подбор на курсовете, системата извършва последователност от проверки и взема курсовете, подходящи за дадената специалност. След това ги предоставя на потребителя, за да извърши той селекцията. При избор на курс системата извършва проверки за това- какви и колко на брой са избраните досега дисциплини. Ако техният брой е равен на

5 (това е максималният брой курсове, които студент може да запише за текущия семестър), потребителят е уведомен чрез съобщение от системата, че е достигнал допустимия лимит и процесът се приключва. Ако броят им е по-малък от 5, системата извършва проверка за удовлетворяване на съответните за дисциплината входни изисквания за компетентности (предишен опит, успех и т.н.):

-Ако изискванията НЕ са покрити, потребителят е уведомен чрез съобщение от системата, че операцията по записването е неуспешна и го приканва да избере дали да продължи със селекцията на други дисциплини, или да приключи процеса. Ако отговорът е НЕ- процесът приключва. Ако той обаче е ДА, системата препраща потребителя отново към опцията за избор на курсове.

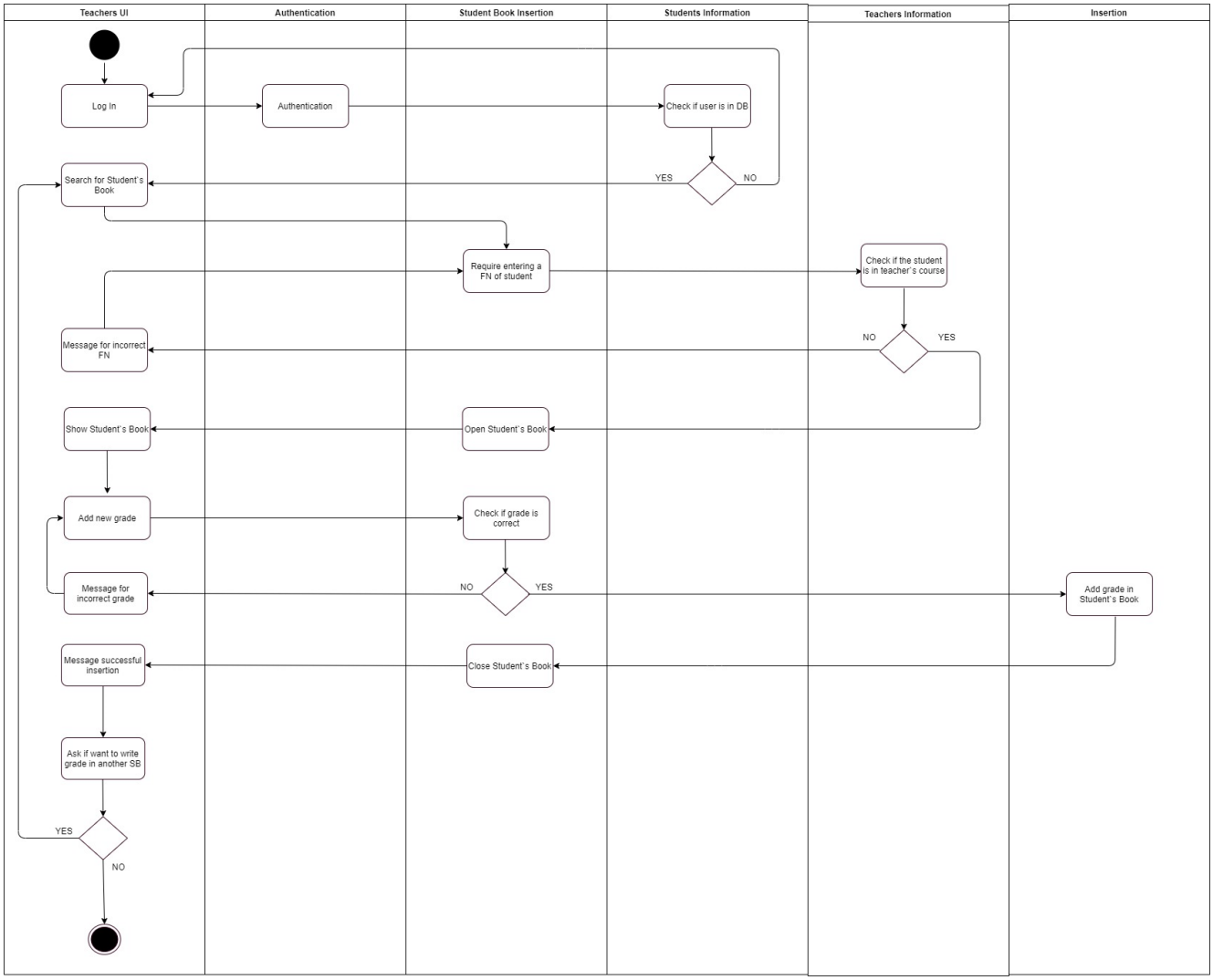
-Ако изискванията СА покрити, студентът бива записан в базата данни за съответния курс. Потребителят е уведомен чрез съобщение от системата, че операцията по записването е успешна и го приканва да избере дали да продължи със селекцията на други дисциплини, или да приключи процеса. Ако отговорът е НЕ- процесът приключва. Ако той обаче е ДА, системата препраща потребителя отново към опцията за избор на курсове.

FP3.3. Описание на обкръжението

В така описания процес не се наблюдава кореспонденция на системата с някоя от външните системи, с които по принцип си взаимодейства.

SP3.Структура на процесите: Вписване на оценка в студентска книжка

SP3.1. Първично представяне



Процесът по записване на изибраема дисциплина от студент е визуализиран благодарение на горната диаграма на активностите. Чрез нея се показва как е постигнато едно от функционалните изисквания към системата, а именно:

> *Системата да поддръжа електронни студентски книжки, които са част от студентския профил. В тях, преподавателите внасят оценките на студентите по записаните от тях дисциплини.*

По-общо, заключенията, които биха могли да се направят чрез недотам задълбочен анализ, са следните:

1) Потребителският интерфейс е разбираем и интуитивен, тъй като потребителите лесно могат да открият необходимите функционалности, за да извършат исканите от тях операции. В допълнение, системата е ангажирана и с дейността по уведомяване на потребителите при всякакви проблеми, възникнали по време на изпълнение, с което ги държи информирани.
2) Бизнес логиката е свързващото звено между това, което потребителите виждат, и конфигурационните файлове за връзка с базата от данни. В нея се извършват всевъзможни проверки с цел покриване на всички случаи.
3) В конфигурационните файлове, служещи за връзка с базата, се извършват подходящите за текущия процес заявки към базата.

SP3.2.Описание на елементите и връзките

При желание на преподавател да впише оценка в дадена студентска книжка той, първо, трябва да е влязъл в профила си. При вход в системата се извършва удостоверяване на самоличността на потребителя, което изисква проверка за наличността на данни за него в базата. Ако не присъства в базата данни, се приканва да въведе валидни потребителско име и парола. Интерфейсът за преподавателите предоставя възможност за търсене на студентска книжка по факултетния номер на студент. След въвеждането на ФН се извършва проверка в базата дали съответния студент е в курса на преподавателя. Ако не, преподавателят бива уведомен, че е въвел грешен ФН и бива приканен да въведе ФН отново. Ако да, то системата му отваря съответната студентска книжка и преподавателят получава възможността да въведе оценката, която желае. Системата извършва проверка за валидността на въведената оценка. Ако тя е невалидна,потребителят е уведомен чрез съобщение и бива приканен да въведе оценка отново. Ако е коректна, оценката се записва в базата данни, книжката се затваря и потребителят получава съобщение за успешно извършена операция по въвеждане на оценка. Потребителят също така получава и запитване от системата дали иска

да въвежда още оценки. Ако да, той е препратен отново към етапа на търсене на съответната книжка по факултетен номер. Ако не, операцията приключва.

SP3.3. Описание на обкръжението

В така описания процес не се наблюдава кореспонденция на системата с някоя от външните системи, с които по принцип си взаимодейства.

4.Архитектурна обосновка

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

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

Едно от важните неща при проектирането на архитектурата на една софтуерна система е правилното идентифициране на различните видове потребители, които ще я използват. Това позволява адекватното определяне на техните права за достъп, използване на дадени функционалности на системата, модифициране на информация и т.н. Това функционално изискване- за няколкото вида потребители, е удовлетворено чрез разделянето на функционалността на системата на отделни модули, осигуряването на модул за всеки вид потребител (В **UI, Business Logic**, **Data Base Configuration Files**) и дефинирането на различни права за достъп и употреба на различните модули.

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

Това функционално изискване се удовлетворява от така създадената архитектура благодарение на модулите **Students UI**, **Teachers UI**, **Students**

Information и **Teachers Information** в **Data Base Configuration Files**. Те се управляват от администраторите на системата, които получават достъп до нея през обособения за тях интерфейс **System Administartors UI**.

4. 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.

Това функционално изискване е изпълнено чрез осигуряването на модула **Accountants UI** в **Administration UI**, **Finance (Salaries, Students Fees, Other Payments)** в **Data Base Configuration Files.** Контролът на финансовите операции се осъществява по следния начин: При необходимост да се направят справки, отчет на парите и т.н., счетоводителите на университета, които имат достъп до системата чрез обосбения за тях **Accountants UI**, достъпват данните от плащанията, които се съхраняват в базата данни чрез модула **Finance** в **Data Base Configuration Files.**

5. Студентите могат да записват одобрени курсове само в рамките на тяхната специалност и при условие, че профилът им отговаря на входните изисквания за компетентности за съответния курс.
Това функционално изискване е най-точно представено чрез визуализирането му посредством **Структурата на процесите: Избор на курс**, като към нея се включва и подробно обяснение (**подсекция FP3** на **секция 3** в текущата документация).

6. Студентите могат да генерират различни видове официални справки за студентския си статус: уверения, академични справки и т.н.

Това функционално изискване е постигнато чрез модула **Students UI**, **Official References Creation (Business Logic), Official References Information (DB Config).** Процесът протича по следния начин: В предоставения им интерфейс студентите имат опцията да генерират справка за статуса си (данните, които трябва да бъдат извлечени от базата, се оформят в структура от данни в **Official References Creation**), която се изпълнява, като се осъществи заявка към базата данни посредством модула **Students Information**. След генерирането й тя се запазва в **Official References Information**.

7. Системата да поддръжа 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.

В архитектурата са обособени отделни модули за преглед на и запис в студентска книжка (**Student Book View, Student Book Insert**). Студентите и преподавателите я достъпват чрез интерфейса **Students Book UI**, като имат съответните права, съобразени с техния статус. Вписаните оценки се съхраняват в базата благодарение на модула **Insertion**, а се преглеждат чрез модула **View**, като и двата са част от **Data Base Configuration Files.** Това функционално изискване е най-точно представено чрез визуализирането му посредством **Структурата на процесите: Вписване на оценка в студентска книжка**, като към нея се включва и подробно обяснение (**подсекция SP3** на **секция 3** в текущата документация).

8. 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.

Всеки от потребителите има достъп до чат в своя профил. Цялата функционалност зад него се намира в модула **Event Creation**. История на съобщенията се пази в модула **Events**.

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

Всеки от потребителите има достъп до чат в своя профил. Цялата функционалност зад него се намира в модула **Message Exchange**. История на съобщенията се пази в модула **Message History**.

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

Това функционално изискване се постига чрез обособяване на логика за тях в модулите **Request Creation** и **Request Acception**. Информация за тях се пази в базата данни посредством модула **Request History** в **Data Base Configuration Files.**

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

Това функционално изискване се постига чрез модула **Security**.

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

Това качествено изискване е удовлетворено чрез разпределянето на модулите, които съдържат по-важната за системата функционалност (тоест ще бъдат по-натоварени), на повече от няколко хардуерни машини. При евентуален срив на едната машина, другите ще поемат нейните отговорности, докато тя бъде възстановена. Това се вижда най-ясно на Структурата на внедряването (**подсекция D3** на **секция 3** в текущия документ).

13. 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.

Това функционално изискване се постига благодарение на модула **Outer Systems Connector.**

14. 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.

Това качествено изискване е удовлетворено чрез разпределянето на модулите, които съдържат по-важната за системата функционалност (тоест ще бъдат по-натоварени), на повече от няколко хардуерни машини. Това се вижда най-ясно на **Структурата на внедряването** (**подсекция D3** на **секция 3** в текущия документ).