**Курсова работа по**

**Софтуерни архитектури и разработка на софтуер**

**UniDigit**

**Изготвил: Георги Демирев, 62296**

# 1. Въведение

## a) Обща информация за текущия документ

* **Предназначение на документа**
* Документа служи за запознаване на всички заинтересовани лица с архитектурата на софтуера - UniDigit.
* **Описание на използваните структури на архитектурата.** 
  + Декомпозиция на модулите показва кои са различните модули и подмодули на системата и връзките между тях.
  + Структурата на разположението показва връзката между софтуерните елементи и елементите на околната среда, в която се намира системата по време на разработката или по време на изпълнението.
  + Структурата на процесите показва как се изпълнява даден процес. Кои са компонентите които се използват, връзките между тях, проследява се всяка стъпка от процеса, времетраенето и кои са блокиращите му операции.
  + Структурата на разработката показва точния процес на разработка на приложението.
* **Структура на документа** 
  + Секция 1: въведение в документа
  + Секция 2: описва декомпозицията на системата на модули.
  + Секция 3: описва допълнителните структури на архитектурата
* 3.1: описва структурата на внедряването
* 3.2: описва структурата на разработката
* 3.3: описва структурата на процесите
  + Секция 4: обосновава изборите за архитектурата

## b) Oбщи сведения за системата

Софтуерна система за управление на процесите и студентската информация в един университет. Системата е създанена така, че да бъде приложима за университети по цял свят и лесно да се добавят функционалности, спрямо нуждите на университета.

## c) Терминологичен речник

* Front End, UI, WebApp – потребителския интерфейс на приложението, което потребителя използва.
* Back End (Server) – сървъра който е скрит от потребителя и изпълнява бизнес логиката.
* API – приложение, услуга.
* Микросървис – едно приложение. Затова логиката е разделена на много отделни приложения които си комунират.

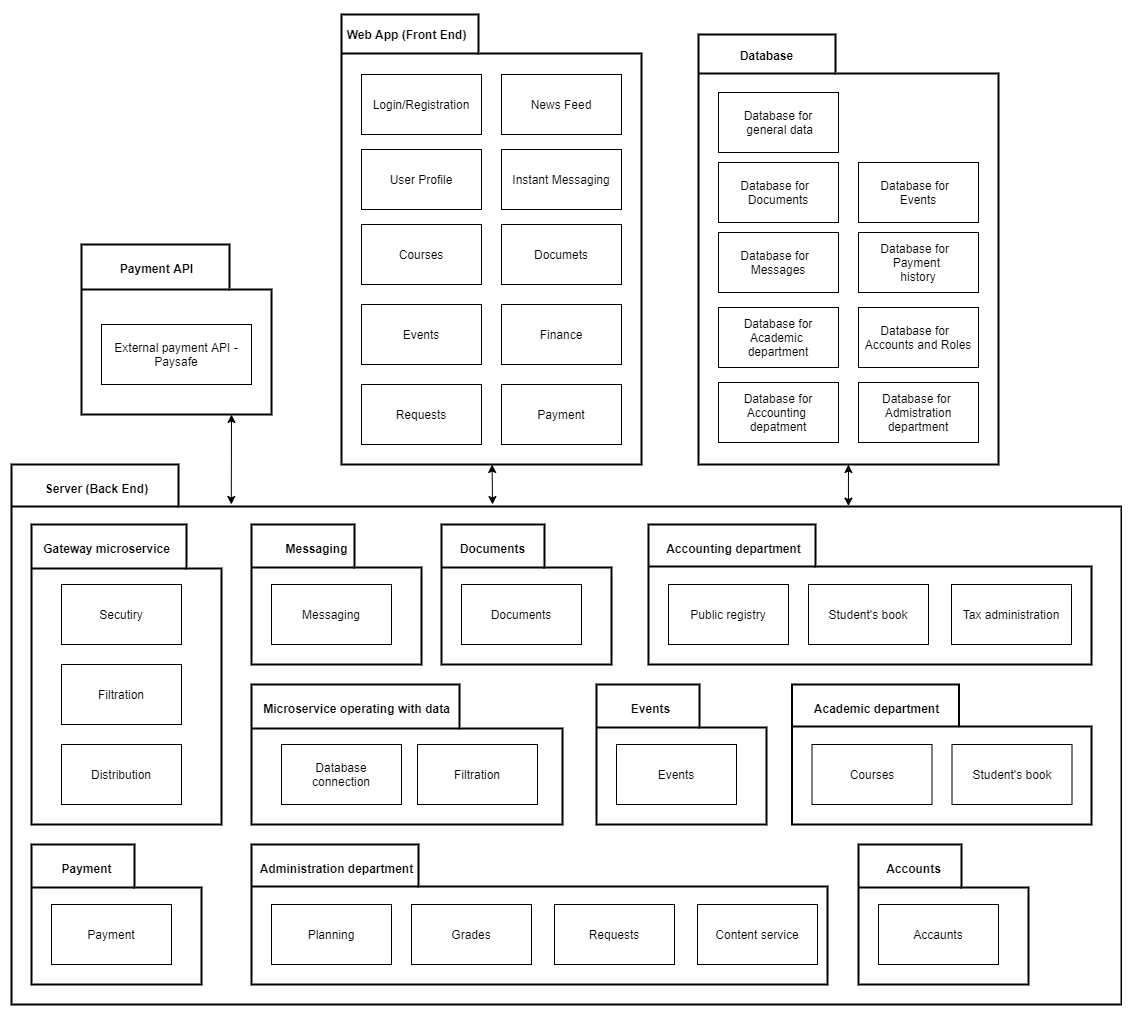
# 2. Декомпозиция на модулите

## a) Общ вид на декомпозицията на модули за системата

Модули

* **Server** - това е модулът който отговаря за бизнес логиката на приложението (back end).
* **Web app** – клиентско приложение с което се осъществява връзката с потребителите (front end) и комуникира със back end-а. Разделено на микросървиси.
* **Database** – сигурна база от данни за съхранение на информацията използвана в системата.
* **Payment API** - интерфейс на външната система чрез която се извършват онлайн плащанията.

## b) Контекстна диаграма



* Диаграмата показва как приложението е разделено на 4 главни модула.
* PaymentAPI e външна система за извършване на плащанията.
* Front End (UI), е потребителския интерфейс, модула с който потребителите комуникират със сървъра и логиката на приложението.
* Server (Back End), е модула който се грижи потребителя успешно да извърши операцията през потребителския интерфейс.
* Database е базата данни, която държи информацията с която сървъра работи.
* Всеки подмодул на сървъра е микросървис който се грижи за отделна логика на приложението.

## c) Подробно описание на всеки модул

Подмодули:

**Server:**

*Разделен на 10 микросървиса отговарящи за различните функционалности, като всеки един от тях си комуникира спрямо нуждата с останалите. Всеки микросървис си има собствена база, като така се получава разделение на данните, без излишни данни за различните функционалности.*

Gateway microservice – приема заявки от web app и ги разпределя към съответните микросървиси

* Security service – всяка заявка идваща от UI, минава първо от тук за да се авторизира и автентикира
* Filtration service – филтрира заявките спрямо ролята на потребителя – студент, преподавател и тн.
* Distribution service – дистрибутира заявките към микросървиса който може да я обработи. Ако например се изпраща съобщение, заявката се препраща към messaging microservice.

Messaging microservice – сървис отговарящ изпращането и получаването на съобщения

* Messaging service – отговаря за изпращането и получаването на съобщения и ги запазва в базата данни

Documents microservice – сървис отговарящ за всички функционалности свързани с документите

* Document service – отгаваря за създаването на различните документи от отдел, който може да създава документ и го запазва в базата. Също отговаря за генерирането и обработка на справки, както и за тяхната истинност.

Events microservice – сървис отговарящ за всички функционалности свързани с събитията

* Events service – отговаря за създаването на публични събития и запазването им в базата

Accounts microservice – сървис отговарящ за всички функционалности на счетоводния отдел

* Accounts service – отговаря за създаване на акаунти и обработката им, както и запазването в базата
  + public bool register(string name, string facultyNumber, string username, string password);
  + private bool checkUsername(string username);
  + private bool checkPassword(string password);
  + public bool login(string username, string password);
  + public User getUser(string username, JWT token);

● Вход login: потребителско име, парола

● Вход getUser: потребителско име, токен, който вече е получен от login

● Вход registration: име, ФН, потребителско име, парола

● Изход: съобщение дали операцията е успешна, и токен който UI използва да оторизира достъпа на потребителя, ако той продължи да използва приложението, след регистрация

Payment microservice – сървис отговарящ за изпращането на заявки към външната система за плащане

* Payment service – отговаря за за изпращането на заявки към външната система за плащане, както и запазването на плащанията в базата

Accounting department microservice – сървис отговарящ за всички функционалности на счетоводния отдел

* Public registry service – отговаря за връзката с държавни публични регистри
* Student’s book service - отговаря за операциите които счетоводния отдел извършва със студентските книжки
* Tax administration service - отговаря за връзката с НАП и данъчната администрация

Administration department microservice – сървис отговарящ за всички функционалности на административния отдел

* Planning service – отговаря за предложения за планове и програми и тяхното одобрение
* Grades service – отговаря за нанасянето и промяната на оценки
* Requests service – отговаря за създаването и одобряването на искания
* Content service – отговаря за учебното съдържание като може да е свързано с външни системи като moodle

Academic department microservice – сървис отговарящ за всички функционалности на учебния отдел

* Courses service – отговаря за записването/отписването на студент за определен предмет
* Students’s book service – отговаря за операциите които учебния отдел извършва със студентските книжки, като позволява само преглед без промяна от студента

Мicroservice operating with data– сървис който отговаря за данните на другите микросървиси и е като централна база на приложението. Всеки друг микросървис го достъпва.

* Database connection service – прави връзката с базата и всички операции към нея
* Filtration – прави филтрация на данните и ги персистира/взима в/от различните бази отговарящи за различните микросървиси

**Web app:**

* Login/ Registration– страница със форми за логин и регистрация.
* News feed – страница на която ще се публикуват събития.
* Instant messaging - потребителски интерфейс за пращане на моменти съобщения.
* User profile – профил на потребителя където ще може да се преглежда информация като лични данни, статут, студенстка книжка и др.
* Finance page – интерфейс за контрол на финансовите операции, управлявани от accounting department, като възнаграждения и такси и разплащане с външни системи. Тук се включва и връзката с външните системи като НАП.
* Courses page – интерфейс за записване и отписване на предмети и създаване
* Documents page – интерфейс за генериране на справки и други документи и верификация на тяхната автентичност
* Events page – интерфейс за управление и създаване на събития
* Requests page – интерфейс за създаване на искане и неговото одобрение от съответния отдел
* Payment interface – интерфейс за плащане

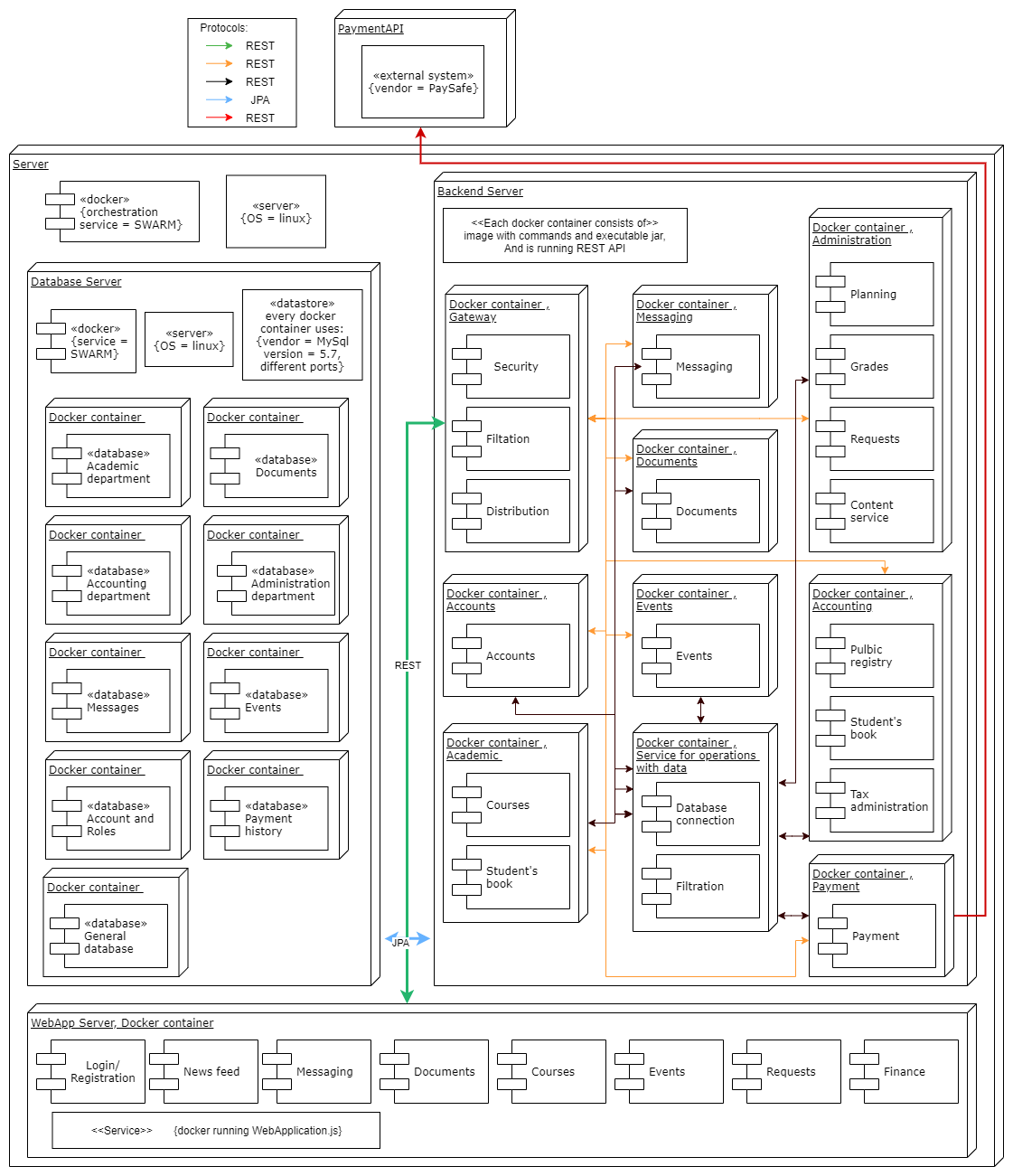
**Database**:

Всеки микросървис има собствена база данни до която другите микросървиси нямат достъп, това се прави с цел всеки микросървис да се грижи за собствените си данни.

# 3. Описание на допълнителните структури

Deployment structure

Структура на внедряването



* Цялото приложение е разположено на виртуална машина с операционна система Linux. По този начи приложението лесно може да се скалира на различни сървъри и cloud.
* На машината има вървяща докер машина и docker swarm за оркестрацията на контейнерите. Всички контейнери от backend + тези от frontend + тези от базата са стартирани на виртуалната машина и биват оркестрирани от swarm.
* PaymentAPI е външен софтуер за извършване на плащанията който поддържа системата.
* Контейнерите са разделени на модули(раздели) които ги обединяват, както се виджа база, backend, frontend. Вътрешно модулите комуникират както е показано със стрелките.
* WebApp-(front end) комунира чрез rest с backend server-а. Той изпраща заявки на gateway микросървиса. Всеки модул на webapp-a отговаря на микорсървисите и gateway микросъвиса разпределя заявките към съответния модул.
* Всяка заявка от webapp към backend-сървъра, минава първо през Security service-а на Gateway микросървиса, като се автентикира и авторизира.
* Всеки микросървис си има собствена база от данни, като има един микросървис, който е главната база на проекта, държащ данни които отговарят за другите микросървиси.
* От връзките със стрелките се вижда с какъв протокол комуникират модулите и кой към кой има връзка. Черните стрелки се пресичат, като показват, че всеки микросървис си комуникира с базата и базата с него. Оранжевите стрелки се пресичат, като показват, че gateway микросървиса си комуникира с всички останали, без базата. Това е направено с цел допълнителен слой защита.
* Микросървисите комуникират с базата чрез JPA.
* Показани са също стартираните файлове в контейнерите.
* Сървъра може да бъде раположен на всякаква операционна система която може да стартира виртуална машина с линукс.

Development structure

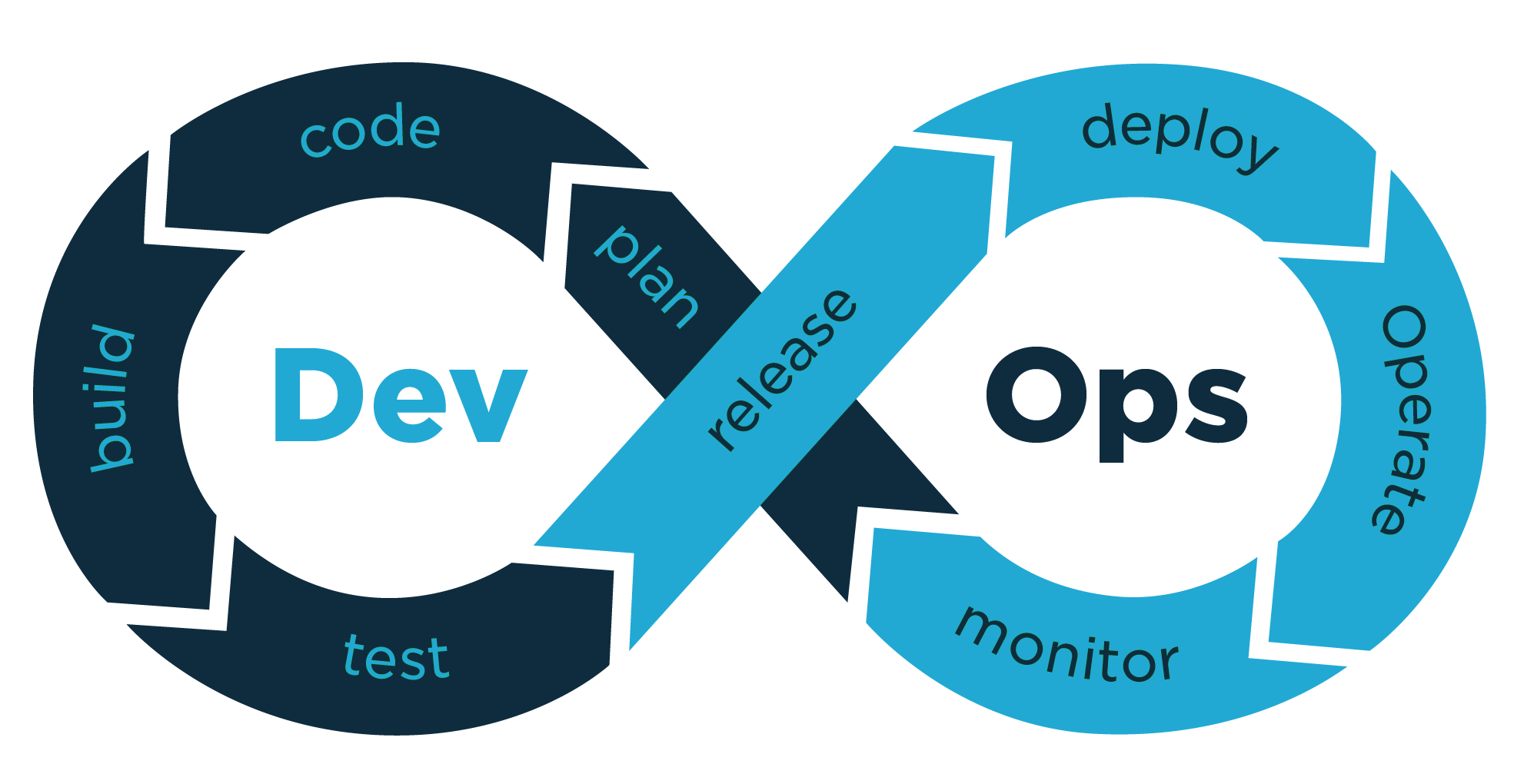
Структура на разработката

Тази структура спестява време на разработчиците, като им показва точния процес на разработката в екип, бързото тестване и доставяне на продукт.

За разработката ще използваме DevOps стил.

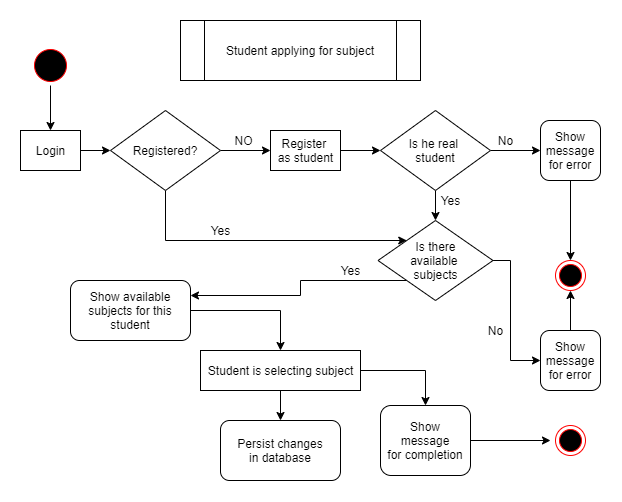
Следващата картинка показва цикъла на DevOps стила за разработка. Затворен agile процес при който всичко е автоматизирано и улеснено за спестяване на време. Тук се използват много различни технологии при всяка стъпка. Ще използваме и continuous integration метода. Това са някои от технологиите които ще ни помогнат в разработката по DevOps стила. По-долу е картинката която показва процеса.

1. Plan – Jira
2. Code – Git, JetBrains, HashiCorp
3. Build – Maven, Bitbucket
4. Test – JUnit, Sonarqube - за постоянна инспекция и автоматично тестване(continuous inspection)
5. Release – Jenkins, за continuous integration метода
6. Deploy - Docker
7. Operate – Swarm ( оркестратор за микросървърната архитектура която използваме в backend-а)
8. Monitor – Elastic stack (elastic search, kibana), Logging.

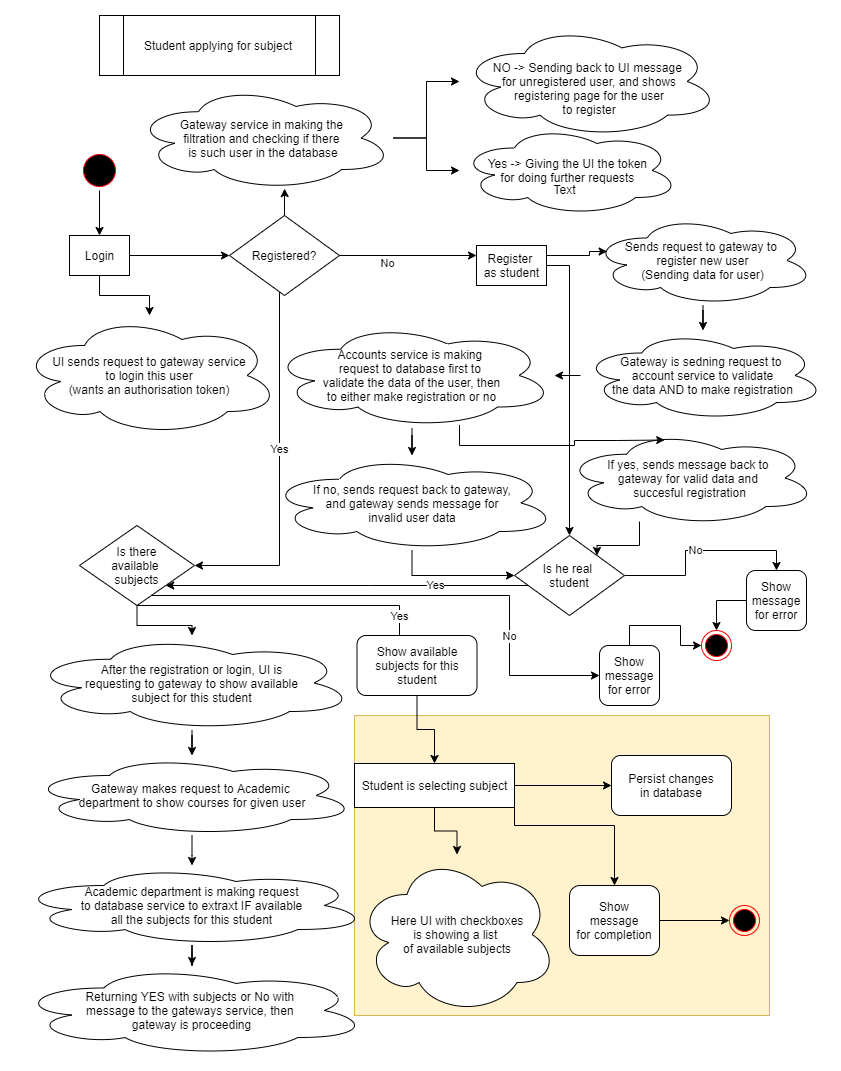


Стуктура на процесите

Процес на избиране на избирателни предмети от студент



* Това е много проста диаграма на последователността на действията при избиране на предмет.
* Със следващата диаграма показвам логиката в Back End сървъра.
* ***Оградените със жълто процеси, не са обяснени изцяло, понеже са аналогични на горните обяснения***.



# 4. Архитектурна обосновка на изискванията

1. Системата обслужва следните отдели в университета:

* Учебен отдел
* Счетоводен отдел
* Студентски съвет
* Административен отдел

*-Всеки отдел си има собствени права, които са раздадени на потребителите на отдела при регистрацията. Тогава след като е влезнал потребителя, той може да използва приложението с функционалностите които неговия отдел предлага. Това се контролира от gateway микросървиса.*

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

*-Всички типове потребители се обработват от account manager и техните роли в приложението биват филтрирани от gateway миркосървиса.*

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

*-Предложения се подават в Planning service, той ги изпраща към съответния отдел за одобрение. Целият процес е: Създава се предложението от преподавател, то се изпраща към базата на planning, след това се изпраща и заява към административния отдел, запазва се и в тяхната база и тогава когато потребител от администрация влезе в приложението, ще има известие за искането от преподавател.*

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

*-Planning service изпраща заявки към учебния отдел и административния отдел получава известия за одобрение или отхвърляне на програми.*

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

*-Accounts service се грижи за всичко свързано с акаунтите, съхранението в базата, регистрацията и всички привилегии на потребителите.*

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.

*-Tax administration сървиса отговаря за всички финансови операции, пази ги в базата, а* *разплащането с външни услуги става чрез PaymentAPI (външна система)*

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.

*-Показано е със структурата на процесите, като за тази функционалност отговаря Courses* *сървиса.*

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

*-Documents service се грижи за всички създавания на документите*

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.

*- Documents service се грижи за верифицирането на документ при началното му създаване и* *при последващото му използване, като има услуга която верифицира документа по снимка.*

10. 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.

*-Student’s book service се грижи за студентските книжки, като той има две имплементации в* *различните отдели на системата. Всеки отдел се грижи за своята работа със студентската* *книжка.*

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.

*-Events микросървиса се грижи за всички събития, тяхното създаване, премахване,* *одобрение.*

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

*-За тази функционалност се грижи messaging микросървиса.*

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

*-Това се случва както е показано на структурата на процесите. Прави се от Planning сървиса.*

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

*-Системата е създадена така, че да има няколко защитни слоя. Основния от които е security* *service във gateway микросървиса.*

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

*-Events service отговаря за тази функционалност*

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

*-Swarm се грижи за това, контейнерите със стартираните приложения, да са винаги* *налични. При отказ на някой сървис, Swarm го рестартира за секунди. С микросървисната* *архитектура, при отказ на някоя част на приложението, не спира цялото приложение, а само една малка част от него, и то само за секунди. А при по-голям проблем с логиката, може да се спре само единия сървис, като това няма да прекъсне работата на останалите.*

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

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

*-Tax administration, Content service и Public registry service правят заявки към външни* *системи чрез REST протокол. В микросървиса винаги може да бъде добавен нов сървис,* *който да отговаря за нова 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.

*-Toва изискване се покрива със услугата Swarm – оркестрация на контейнери. Когато се* *претовари един микросървис, Swarm стартира още един който поема натоварването.*

**Изготвил: Георги Демирев – 62296, 2ра група, 2ри курс**