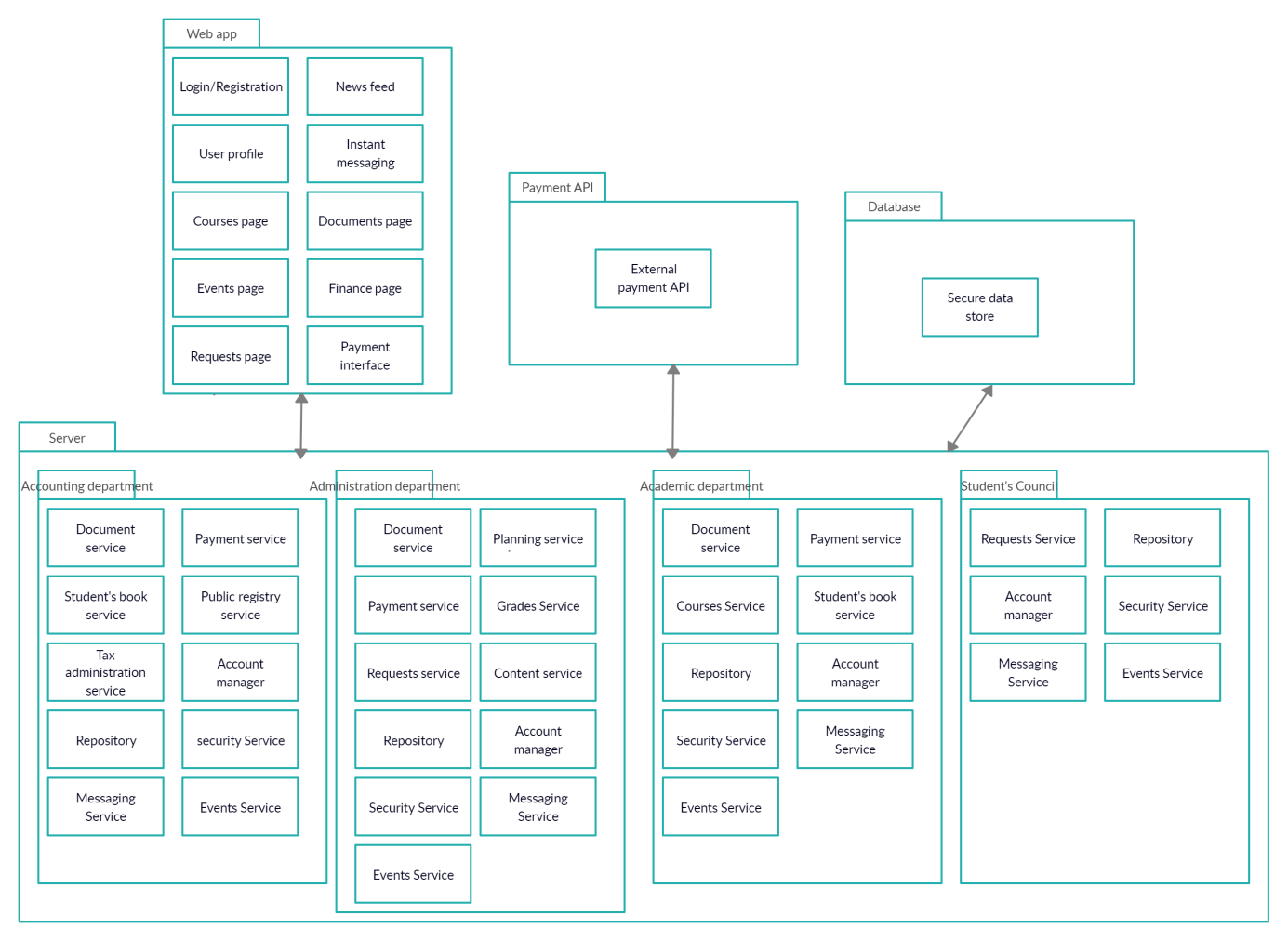
Домашно 2

***Модулна декомпозиция***

******

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.

-**Database** – сигурна база от данни за съхранение на информацията използвана в системата.

**-Payment API** - интерфейс на външната система чрез която се извършват онлайн плащанията.

Подмодули:

**Server:**

*Разделен на четири микросървиса отговарящи за четирите отдела и различните им функционалности, като всеки един от тях си комуникира спрямо нуждата*

*Коментар: преподавателите са част от administration department*

Accounting department – сървис отговарящ за всички функционалности на счетоводния отдел

* Document service – отговаря за обработка на справки, както и за тяхната истинност
* Payment service – контролира финансовите операции които засягат другите отдели и потребители
* Student’s book service – отговаря за заверки на книжките
* PublicRegistry service – отговаря за връзката с държавни публични регистри
* TaxAdministration service - отговаря за връзката с НАП и данъчната администрация

Administration department – сървис отговарящ за всички функционалности на административния отдел

* Document service – отговаря за генерирането и обработка на справки, както и за тяхната истинност
* Planning service – отговаря за предложения за планове и програми и тяхното одобрение
* Payment service – отговаря за възнагражденията които идват от accounting department
* Grades service - отговаря за нанасянето и проманя 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 – сървис отговарящ за всички функционалности на учебния отдел

* Document service – отговаря за генерирането на справки, уверения, както и за тяхната истинност
* Payment service – отговаря за таксите които идват от accounting department
* Courses service – отговаря за записването/отписването на студент за определен предмет
* Student’s book service – отговаря за студентската книжка на студента, като позволява само преглед без промяна от студента

Student council – сървис отговарящ за всички функционалности на студенския съвет

* Requests service – отговаря за създаването на различни искания

*Всеки от четирите отдела имат следните сървиси с различни характеристики:*

*(ПРИМЕР за account manager: направен е за всеки от четирите отдела, защото всеки отдел се нуждае от различен account manager, защото има различни потребители които използват сървиса – студент, член на административен отдел и тн.. , по този начин приложението се разбива на по-големи по логика модули, които си комуникират и лесно се поддържат и променят)*

* Account manager - който отговаря за профилите на съответните потребители и работи с данните им.
* Security service – отговаря за автентикацията и авторизацията със съответните филтри за всеки отдел и за защитата на данните от неоторизирани лица
* Events service – отговаря за създаването на публични събития
* Messaging service – отговаря за личните съобщения между потребителите
* Repository – връзката с базата данни за всеки отдел

**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.

Courses page – интерфейс за предоление на учебни планове

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

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 удобрение от съответния отдел

Payment interface – интерфейс за плащане

**Database**:

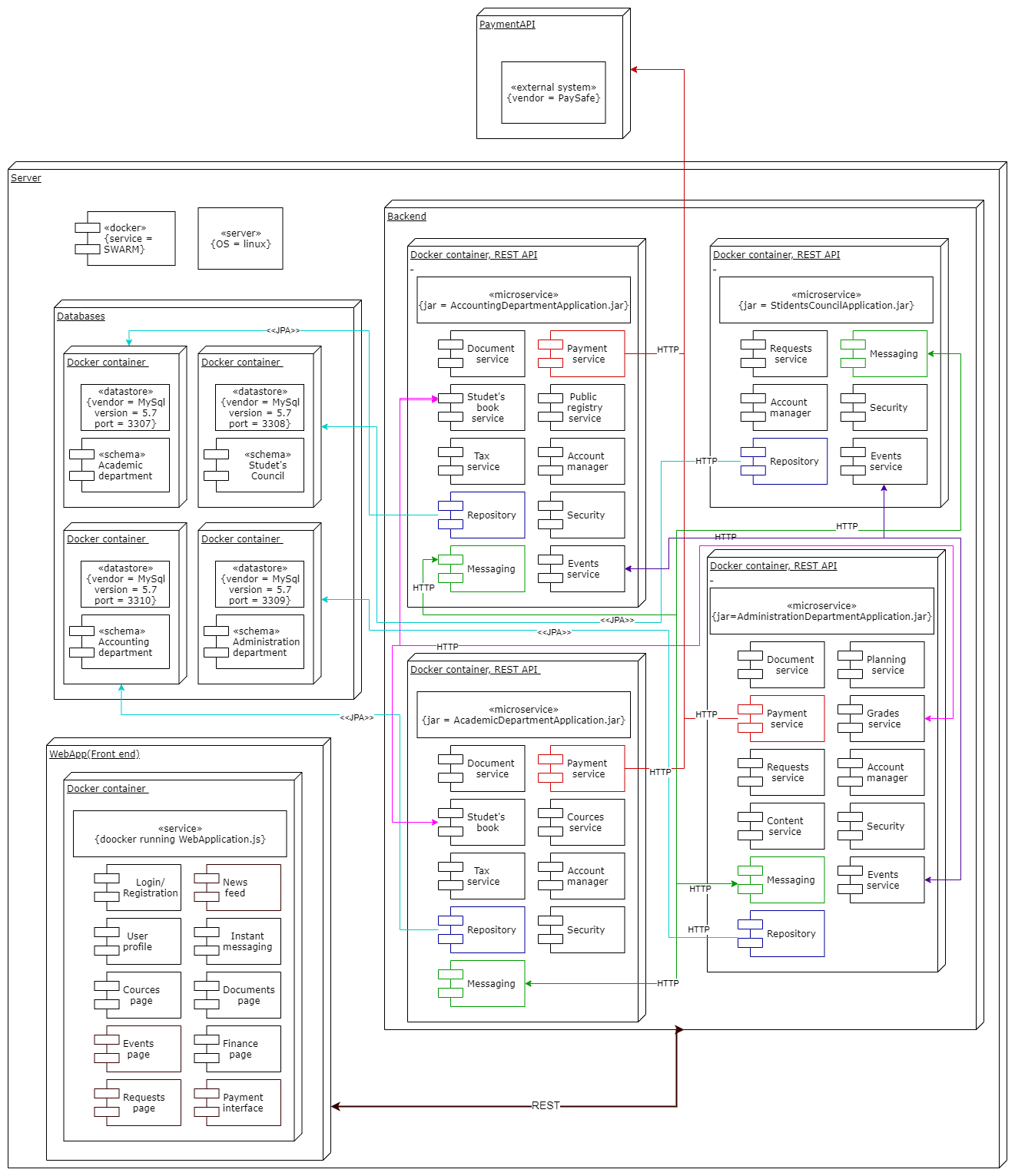
Всеки отдел има собствена база данни до която другите отдели нямат достъп, това се прави с цел всеки отдел да се грижи за собствените си нужди.( Например потребител в административния отдел няма факултетен номер.)

Домашно 3

Допълнителни структури на архитектурата

**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, backedn, frontend. Вътрешно модулите комуникират както е показано със стрелките.

-WebApp-(front end) комунира чрез rest с backend server-а. Всеки негов модул отговаря на модулите в микорсървисите. Не съм направил стрелки, защото стават много оплетени и се повтарят. Например Login/registration модула на webapp комуникира с всеки account manager и security модул на 4те микросървиса. User profile отново комуникира с account manager, като задължително се авторизира пред security(всеки модул се авторизира пред security). Payment interface, например, не комуникира с отдел студенти, защото те нямат тази услуга и не им е нужна. Респективно всички останали модули на webapp комуникират с backend модулите, за техните нужни.

-Всеки вътрешен модул на всеки докер контейнер комуника с останалите модули в контейнера в който се намира. Например account manager -> repository.

-Всяка заявка от webapp към backend-сървъра, независимо кой от 4те микросървиса, минава ПЪРВО през Security service-а на rest api-то, като автентикира и авторизира.

-Подмодула messaging на всеки микросървис от backend-а, комуникира с останалите микросървиси по rest. Аналогично се вижда че repository комуникира само в една посока към собствената си база

-Единствените повтарящи се данни в 4те бази, са тези за events service и messaging, понеже те твябва да знаят за другите събития.

-От връзките със стрелките се вижда с какъв протокол комуникират модулите и кой към кой има връзка. Когато няколко стрелки се пресичат, означава че всеки комуникира с всеки.

-Микросървисите комуникират с базата чрез JPA (java persistent api).

-Показани са също стартираните файлове в контейнерите.

**Development structure**

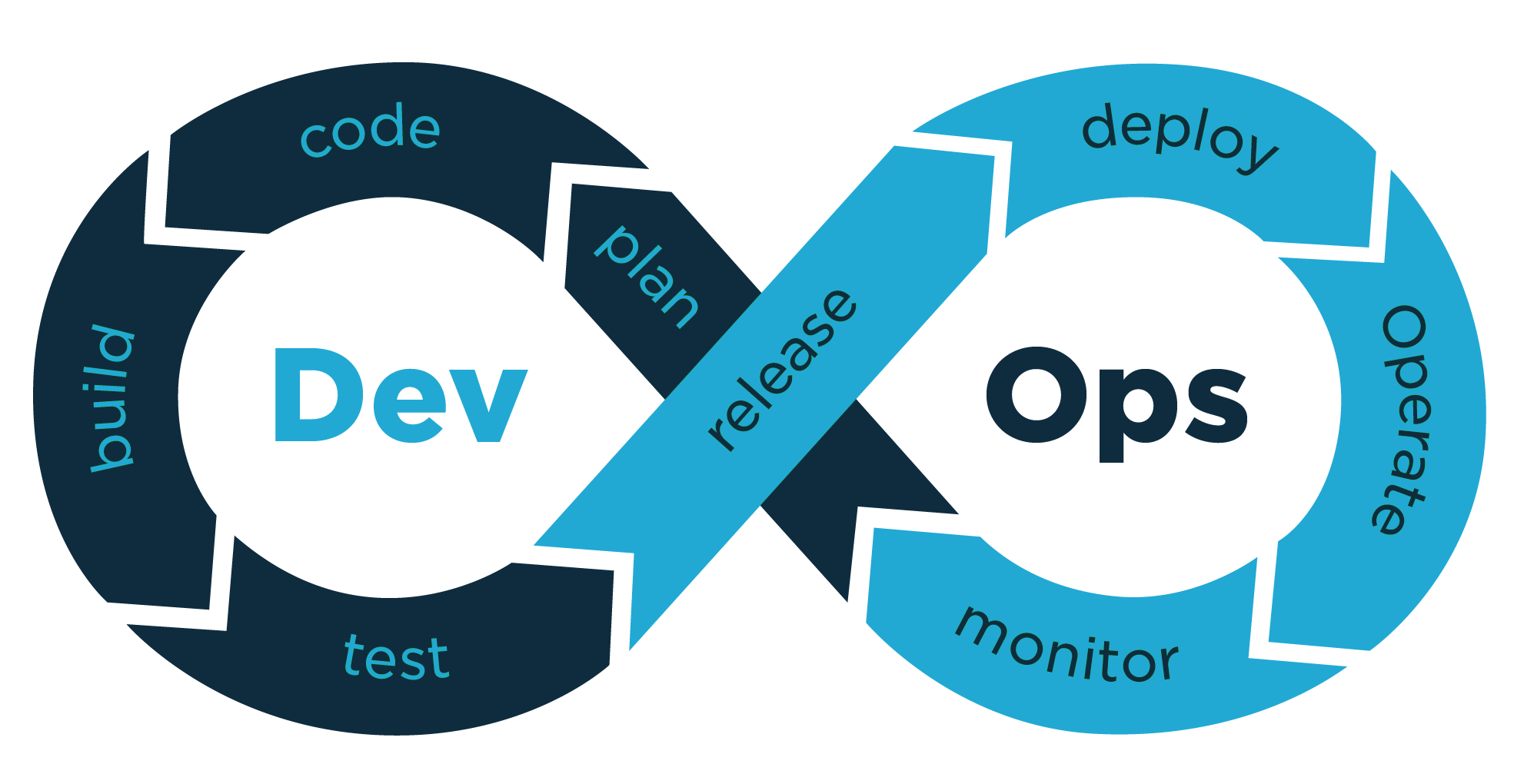
**(Структура на разработката)**

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 технологиите които ще ни помогнат в разработката по DevOps стила. По-долу е картинката която показва точния процес.

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.



ПП: Останах сам по проекта и не ми стигна времето да направя още една структура. Предварително бях подготвил тази за разработка.

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