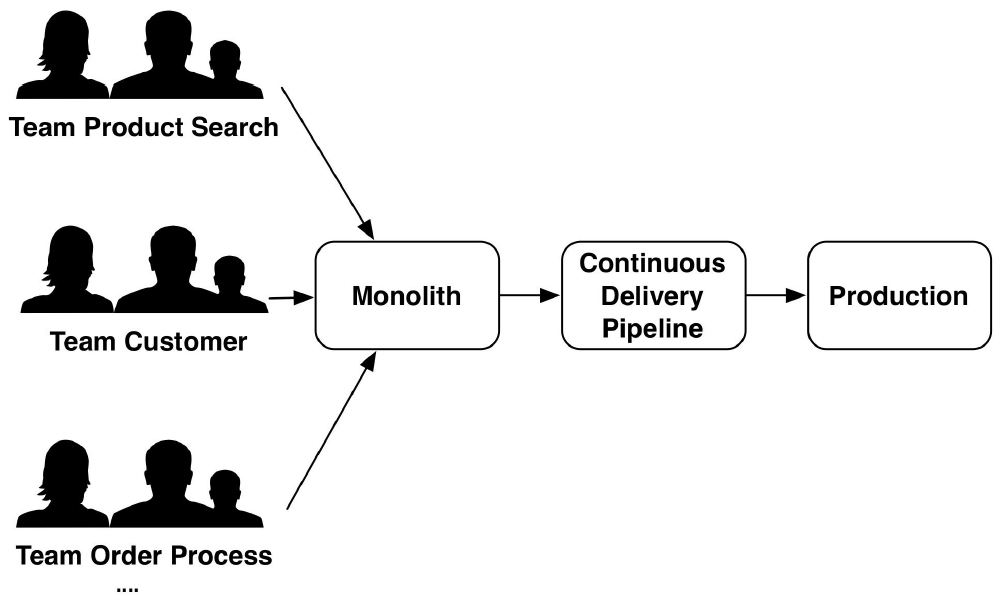
**MicroServices**

*Упражнение*

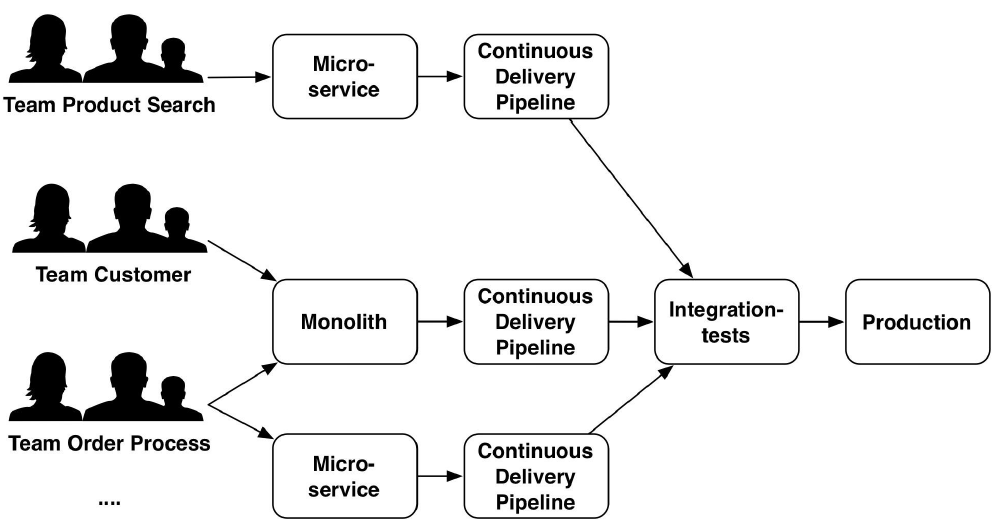
Една компания ръководи магазин за електронна търговия, който е основният източник на приходите на компанията. Това е уеб приложение, предлагащо много различни функции като регистрация и администрация на потребители, търсене на продукти, преглед на поръчките и процеса на поръчка.

Това приложение е *монолит*: може да бъде внедрено (*deployed*) само в неговата цялост. Всеки път, когато дадена функция се промени, цялото приложение трябва да бъде внедрено наново.



*Различни екипи при монолит*

Взема се решение монолита да се раздели на по малки *микроуслуги*. Всяка една от тях реализира една функция (като търсенето на продукти) и се разработва от отделни екипи. Всеки екип носи пълната отговорност за нея, като се започне от проектирането до внедряването.



*Различни екипи след промените*

**Задача**

**А)** Кои са най важните предимства относно това решение? Оценете всяко едно от следните в таблицата (*1 = няма реално предимство, 5 = много голямо предимство*). Обосновете кратко вашият отговор.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *1 = няма реално предимство*  *5 = много голямо предимство* | | | | |
|  | **1** | **2** | **3** | **4** | **5** |
| *Strong modularization* |  |  |  |  | X |
| *Reusability* |  |  |  | X |  |
| *Scalability* |  |  |  |  | X |
| *Technology independent* | X |  |  |  |  |
| *Service abstraction* |  |  |  | X |  |
| *Faster development* |  |  | X |  |  |
| *Team independence* |  | X |  |  |  |

Високата модулярност (strong modularization) се изразява в това, че ако системата е изработена, използвайки различни технологии и езици като Ruby, Java, .NET, то нежелани зависимости (dependencies) лесно може да се появат. Но при микросървисите комуникацията се осъществява посредством леки протоколи като HTTP и REST услуги и по този начин трудно може да възникнат някакви зависимости. Преизползваемостта (reusability) с оценка 4 е едно от предимствата на този архитектурен стил, тоест лесно може да се преизползва бизнес логика в различните сървиси. Мащабируемостта (scalability) с оценка 5 е съществено предимство, защото микросървисите допринасят именно с това, че при увеличаване трафика на потребителските заявки, те лесно могат да бъдат конфигурирани, така че да обработят този поток. Независимост от технологиите (technology independent) е с оценка 1, защото както при високата модулярност могат да настъпят сериозни проблеми при взаимотношенията на различните използвани технологии. Също така различните връзки между микро услугите са скрити и не се знае кой с кой сървис комуникира, което може да доведе до значителни проблеми. Абстрактност на сървисите (service abstraction) е с оценка 4, защото до известна степен всеки сървис носи солидна доза абстрактност с цел преизползване на кода. Бързото развитие на системата (faster development) има оценка 3, защото микросървисите имат много компоненти, които трябва да бъдат деплойнати, контролирани и изтествани. Следователно се увеличава сложността на операциите и използваните инфраструктури от системата. За да се предотврати това, микросървисите трябва да осигурят, че тези операции са автоматизирани и по-този начин системата може да се поддържа и развива сравнително бързо. Независимостта на екипа (team independence) е с ниска оценка, защото трябва да се поддържа много добра координация например между разработчиците, които отговарят за обработката на поръчките в е-магазина и тези, които отговарят само за прегледа на поръчките. От съществено значение е също дали разработчиците са правилно разпределени за конктретните задачи при изработката на системата.

**B)** От друга страна, освен позитивите, кои са предизвикателствата, които трябва да се очакват при преминаването към архитектурата на микроуслуги? Кое от тези предизвикателствата крие най-голям риск според вас и защо?

* Скрити връзки

Архитектурата на системата съдържа различни връзки между сървисите. Както споменах по горе, не е очевидно кой с кой микросървис комуникира, което от своя страна носи големи затруднения за разработчиците.

* Рефактуриране

Високата модулярност въпреки своите предимства носи и някои недостатъци, като рефактурирането на кода, т.е пренасянето на логика от един сървис към друг.

* Поддръжка

Поради наличието на много компоненти и зависимостти, поддръжката на системата и CI (continuous integration) са сериозни предизвикателства за разработчиците. Затова е нужна сериозна автоматизация на процесите.

* Дистрибутивна система

Разработчиците са изправени пред увеличена сложност на системата сравнение с монолитната архитектура. Комуникацията между микросървисите може да доведе до грешки поради проблеми с интернета. Различните извиквания (calls) са по-бавни в сравнение тези при монолита.

**C)** Кои от предимствата на този архитектурен стил (*MicroServices*) са от по голямо значение, от гледна точка на следните заинтересовани лица (*stakeholders*):

* архитект.
* разработчик.
* ръководител на проекта.
* клиент.
* арихетект – Strong modularization, Scalability, Technology independent
* разработчик – Reusability, Service abstraction, Faster development
* ръководител на проекта – всички изброени са от съществено значение
* клиент – Faster development

Причините са подробно описани в т.A и B.

*Забележки*

* Преименувайте файла в формат ***име\_ФН.docx*** преди да го качите.
* За решаването на задачата, ако искате, вместо общия сценарии описан тук, можете да използвате по избор проект след като го посочите.