

Аутоматско тестирање микросервисних апликација

Студент: Никола Димић

Циљ рада

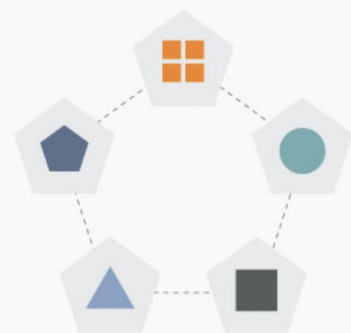
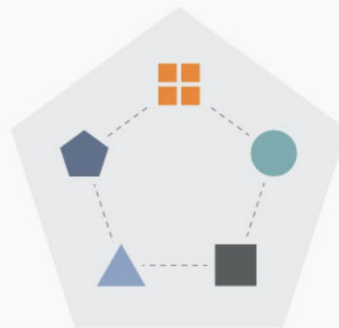
- Приказивање различитих врста и техника аутоматског тестирања микросервисних апликација

Преглед излагања

1. **Микросервисна архитектура**
2. **Имплементација микросервисне апликације**
3. **Аутоматско тестирање**
4. **Врсте аутоматског тестирања**
5. **Имплементација различитих врста и техника аутоматског тестирања**
6. **Резултати тестирања**
7. **Закључак**

Микросервисна архитектура

- Микросервисна архитектура — популаран избор за заснивање дизајна апликација
- Систем — скуп малих сервиса оријентисаних ка специфичном домену који су ниско спрегнути
- Вид дистрибуираних система
- Особине:
 - Композитност (искористивост)
 - Лака интеграција
 - Технолошка хетерогеност
 - Отпорност на отказивање
 - Скалирање



Имплементирана микросервисна апликација

Апликација за претрагу филмова и
књига као и њихових оцена на
најрелевантнијим сајтовима

Клијентска апликација



Микросервиси

- Movies
- Books

Микросервиси

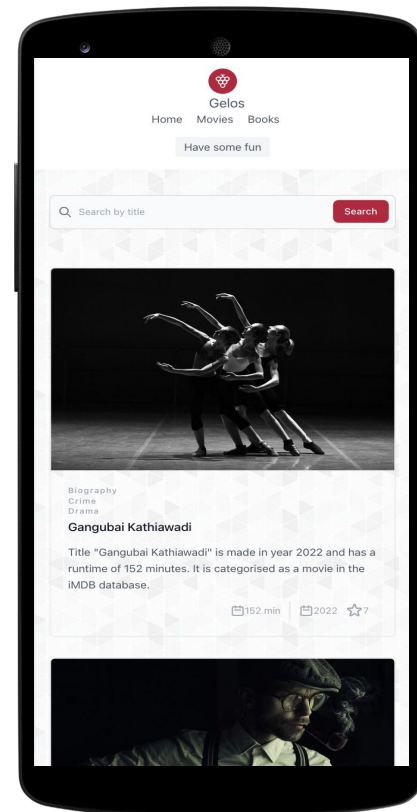
- Комуницирају путем протокола *HTTP* па се лако интегришу
- Сваки има независну базу података која користи *SQL* језик

Микросервис *Movies* — Омогућава баратање подацима о филмовима и оценама филмова са сајта *IMDB*

Микросервис *Books* — Омогућава баратање подацима о филмовима и оценама филмова са сајта *GoodReads*

Клијентска апликација

- Имплементација коришћењем библиотеке *React*
- Састоји се од скупа компоненти
- Комуницира са сервисима коришћењем *HTTP* протокола



Аутоматско тестирање

Тестирање софтвера — процес провере софтвера са циљем проналажења грешака и провере квалитета.

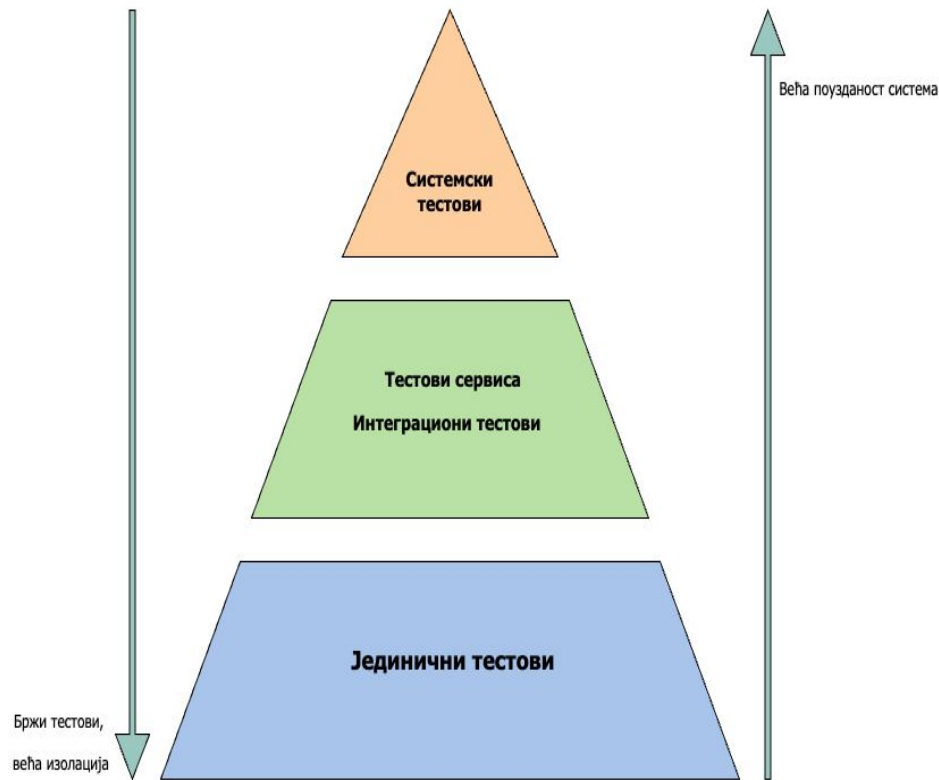
Аутоматско тестирање:

- Убрзава процес тестирања
- Омогућава тестирање већег скупа функционалности
- Омогућава континуално тестирање у току развоја

Није могуће све аутоматизовати.

Врсте аутоматског тестирања

- Јединично тестирање
- Тестирање интеграција
- Системско тестирање





Јединично тестирање

Тестирање појединачних логичких целина у изолацији.

Циљ: Верификовати да се свака од појединачних јединица кода извршава у складу са дефинисаним захтевима

Најбржи и најбројнији

Јединично тестирање:

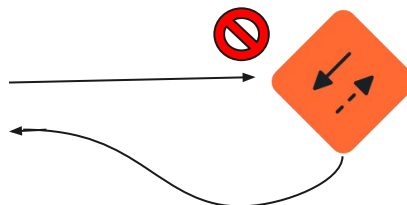
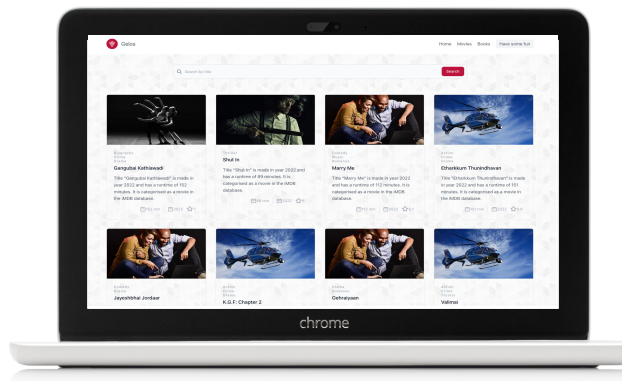
- **Клијентске апликације**
 - Свака од компоненти апликације се тестира изоловано
- **Микросервиса *Books***
 - Свака функционалност сервиса се тестира изоловано
- **Микросервиса *Movies***
 - Свака функционалност сервиса се тестира изоловано

Коришћене технологије:

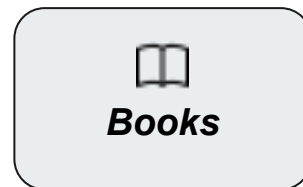
Jest, React Testing Library, MSW

Како осигурати изолованост?

Клијентска апликација



„Лажни”
подаци



Тестирање интеграције

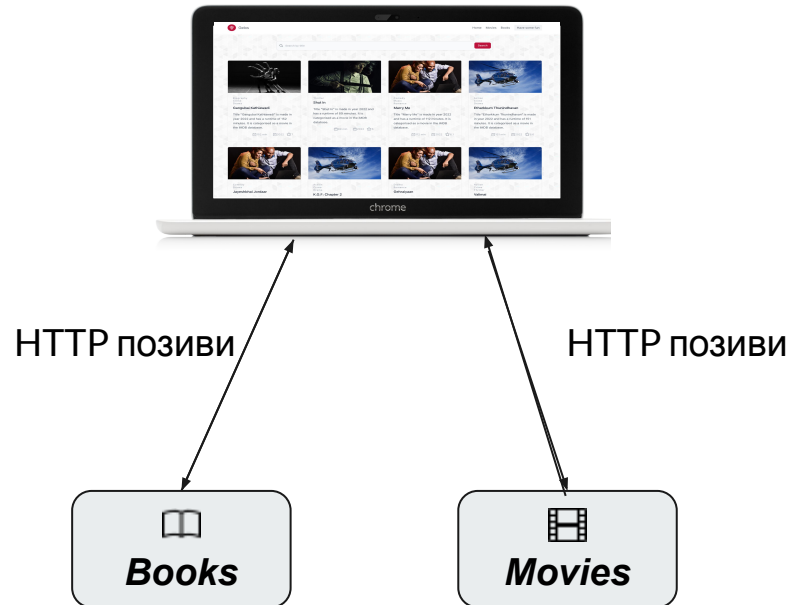
Тестирање комуникације између више јединица кода

Циљ: Верификовати да интеграција јединица кода ради исправно

Брзи. Број тестова зависи од броја интеграција

Верификација интеграције:

- Сервиса *Movies* и клијентске апликације
- Сервиса *Books* и клијентске апликације





Системско тестирање

Тестирање система као целине из угла крајњег корисника у реалном окружењу.

Циљ: Верификовати да систем функционише као целина.

Најспорији

Развојни оквир *Playwright*

Покреће се погон прегледача и симулира корисничко понашање

Елементима на страници приступа се помоћу различитих селектора

Паралелно се извршава више случајева употребе апликације

Тестирати на више типова прегледача, и величина екрана

Резултати тестова

Након извршавања тестова генерише се извештај о успешности

Q		All 13	Passed 7	Failed 6	Flaky 0	Skipped 0
booksTests.spec.js	257ms					
Books api integration tests > Get movies test - no parameters	55ms					
booksTests.spec.js:14						
Books api integration tests > Get books test - page parameter specified	33ms					
booksTests.spec.js:28						
Books api integration tests > Get books test - items per page parameter	50ms					
booksTests.spec.js:44						
Books api integration tests > Get books test - all parameters	50ms					
booksTests.spec.js:59						
Books api integration tests > Get books test - Invalid url	33ms					
booksTests.spec.js:9						
Books api integration tests > Get specific book	36ms					
booksTests.spec.js:70						
movieTests.spec.js	321ms					
Movies api integration tests > Get movies test - no parameters	40ms					
movieTests.spec.js:22						
Movies api integration tests > Get movies test - items per page parameter	50ms					
movieTests.spec.js:55						
Movies api integration tests > Get movies test - Invalid url	33ms					
movieTests.spec.js:17						
Movies api integration tests > Get movies test - page parameter specified	89ms					
movieTests.spec.js:37						
Movies api integration tests > Get movies test - all parameters	73ms					
movieTests.spec.js:72						
Movies api integration tests > Get specific movie	19ms					
movieTests.spec.js:90						
Movies api integration tests > Edit specific movie	17ms					
movieTests.spec.js:105						

Брзина извршавања тестова

Врста тестова	Број тестова	Паралелно извршавање целог скупа	Извршавање једног теста
Јединични тестови - 45 клијентска апликација		0.8-1 s	0.8 s
Јединични тестови - 10 микросервис <i>Movies</i>		1 s	0.4 s
Јединични тестови - 10 микросервис <i>Books</i>		1 s	0.4 s
Интеграциони тестови 16		0.4-2 s	0.4 s
Системски тестови — 23 <i>Chrome</i>		7-9 s	1 s
Системски тестови — 23 <i>Firefox</i>		7-8 s	1 s
Системски тестови — 23 <i>Safari</i>		7-11 s	1 s



Закључак

- Микросервисна архитектура
 - Ниска спрегнутост омогућава различите врсте тестирања
- Аутоматско тестирање - важно на свим нивоима
 - Тестирање јединица
 - Начини за постизање изолације
 - Тестирање интеграција сервиса
 - Тестирање система
 - Стратегија и брзина тестирања

Хвала на пажњи

