SALIEX Julian
BERGER Pierre

TP PERFORMANCE

SNKERS
ESGI.COM
NYTIMES.COM

Application 1:
SNKERS

url: projet local / localhost avec les container lancés

Snkers est une marketplace, les clients sont dans la capacité de pouvoir acheter des “sneakers” directement auprès du site mais ils peuvent également en acheter auprès d’autres vendeurs.

Objectif de l’application:
- Permettre l’achat et la vente de produits
- Mise en relation entre particuliers

Type d’utilisateurs prévus:
- Passionnés de sneakers

Objectif poursuivi par le test de performance:
S’assurer que l’application est apte à répondre dans des temps cohérents aux opérations sur les produits

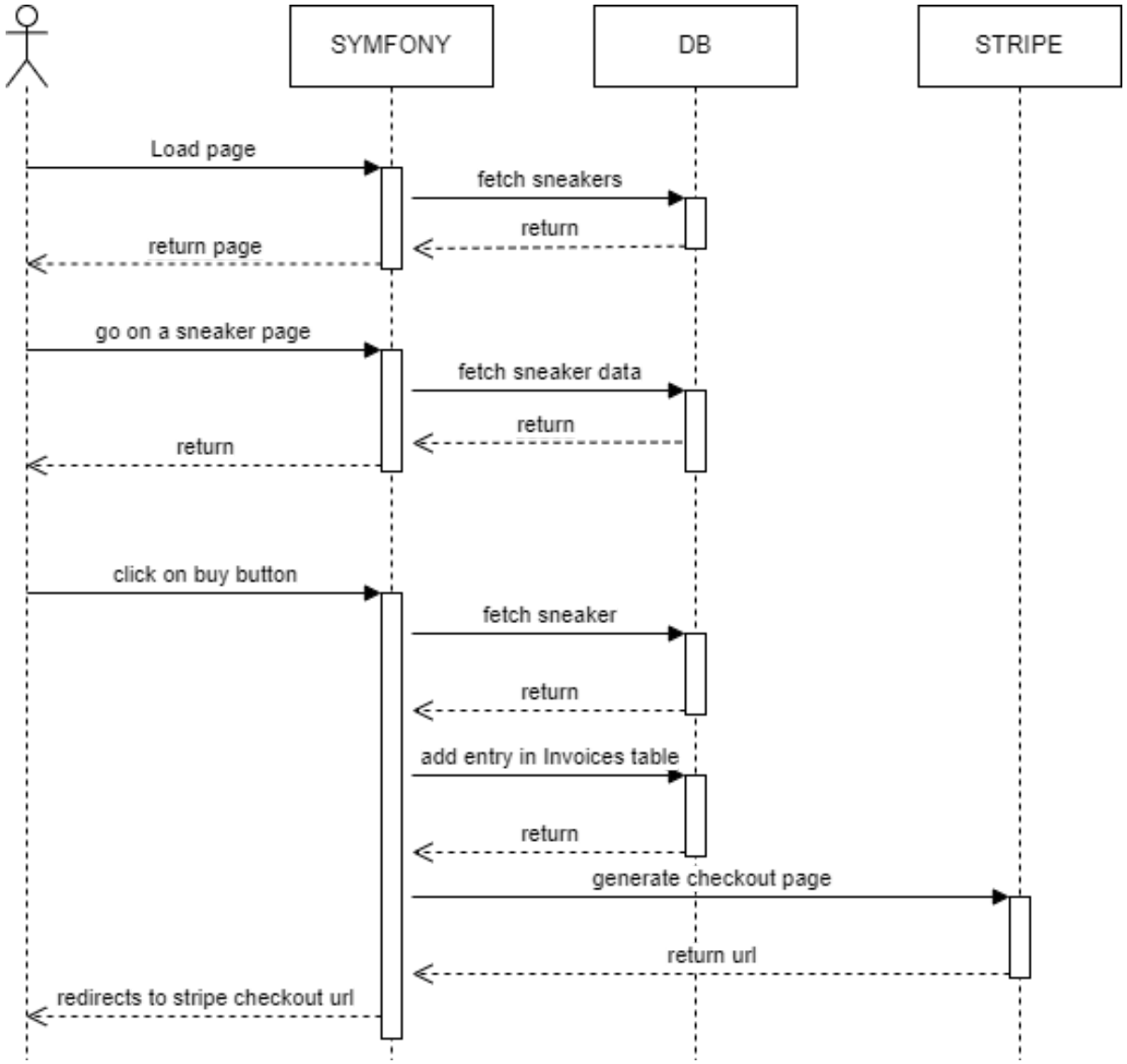
Architecture de l’application

Technologies:
- Symfony
- Postgres:13-alpine
- Twig
- Php 8.0
- Nginx

Services:
- php
- nginx
- postgres

Infrastructure de production:
Solution Heroku

Interactions entre les parties de l’application:



Exigence de test

But du test:
Vérifier la fiabilité de l’application dans une situation de forte sollicitation.

On prévoit 100 utilisateurs quotidien

(On part du principe que les comptes sont déjà inscrit et ont déjà remplis leur informations personnelles)

Business processes	User Load	Response Time	Transactions per hour
Se connecter	40	2s	180
Se rendre sur la page d'accueil affichant les sneakers	40	1s	180
Se rendre sur la page d'un produit en vente	40	1s	180
Se rendre sur la page de paiement Stripe	40	2s	50

Environnement de test

Les tests vont avoir lieu dans des containers docker

CPU	Apple Silicon M1 Pro
Mémoire	16 go
OS	MacOS
Software	Gatling

Environnement de production 2 CPUs
Coefficient: 1 utilisateur en prod = 0.5 en test
50 utilisateurs en environnement de test

Planification des tests

Besoins performatifs de l'application:
L'application a besoin d'être réactive et d'offrir des temps de chargement entre chaque page est inférieur à 2 secondes dans le but d'offrir la meilleure expérience de navigation aux clients.

Métriques pris en compte:
Memory pages/second
Page faults/second
Response time
Thread counts
Hit ratios
Maximum active sessions

Type de tests visés:
Spike Testing
Load Testing
Volume Testing

Étapes des tests

Step #	Business Process Name: navigate to Sneaker payment
1	Login
2	Home Page
3	Select Product
4	Payment Page

Préparation de la données:

Création d'un jeu de donnée d'un seul utilisateur générique ({ name: "test", surname: "test", mail: "test@gmail.com", …})
Utilisation de celui dans un fichier de fixture de symfony dans une boucle et concaténation de l'index d'itération de la boucle dans le mail de

l’utilisateur, ainsi que validation de son compte pour ne pas à avoir à recevoir de mail.

Exemple:

```php
public function load(ObjectManager $manager): void
{
    for($i=0;$i<USER_NB;$++){
        $user = (new User())
                    ->setEmail(`test{$i}@gmail.com`)
                    ->setIsVerified(true)
                    ->setName('test');
        $user->setPassword($this->userPasswordHasher->hashPassword($admin, 'test'));
        $manager->persist($user);
    }
    $manager->flush();
}
```

Exécution des tests

Résultats des tests

Analyses et Optimisations proposées

Application 2:

ESGI.FR

url: https://esgi.fr

Il s'agit du site officiel de l'esgi, il permet d'avoir plus d'informations sur les cursus, les diplômes etc de l'école .

Objectif de l'application:

- Présente l'école
- Indique les formations relatives à l'école
- Mise en contact vers l'école

Type d'utilisateurs prévus:

- Des lycéens et des étudiants

Objectif poursuivi par le test de performance:

Vérifier si l'application peut répondre dans des temps cohérents à un volume important de requêtes d'affichages de pages

Architecture de l'application

Technologies:

- Wordpress

Exigence de test

Environnement de test

Application 2:

NYTIMES.COM

url: https://nytimes.com

New York Times est un quotidien new-yorkais distribué internationalement et l'un des plus grands journaux américains. Son site permet de visualiser ses nombreux articles et de se tenir informer de l'actualité, et il permet aussi de pouvoir y effectuer des recherches.

Objectif de l'application:

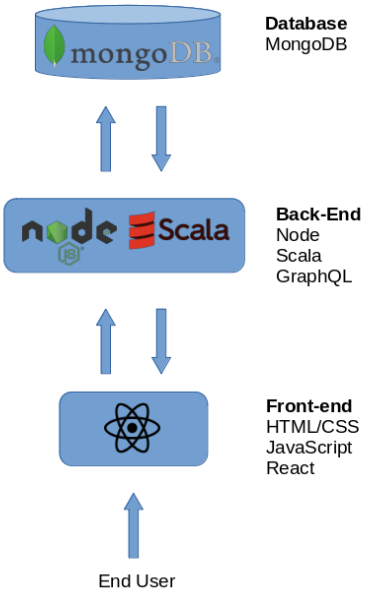
- Informer de l'actualité

Type d’utilisateurs prévus:

- Lecteurs / Internautes journalier

Objectif poursuivi par le test de performance:
Vérifier si l’application peut répondre dans des temps cohérents à un volume important de recherche et d’affichages d’articles NYtimes.

Architecture de l’application



Technologies:

- Stack javascript
- React
- NodeJS
- GraphQL
- Apollo
- Scala

Nous avons décidé d’adopter une architecture monolithique ce qui permet une mise en production plus simple.
Nous hébergeons notre solution sur le cloud avec Heroku.

Exigence de test

Il est obligatoire d’effectuer des tests de charge pour tester la robustesse de l’application car les articles sont publiés de manière journalière, de ce fait, un nombre important de lecteurs vont faire un grand nombre d'appels dans une intervalle de temps très restreint.

On prévoit 300 000 lecteurs journaliers

Business process 1: Se rendre sur la page d’accueil
Business process 2: Se rendre sur la page de connexion
Business process 3: Se rendre sur la page de contact

Business processes	User Load	Response Time	Transactions per hour
1	1900	1.8s	1900
2	500	1.3s	1200
3	200	1s	400

Environnement de test

CPU	Intel i5-8365 U
Mémoire	32 go
OS	Fedora release 36
Software	Gatling, LoadFocus

Nous possédons 2 CPU dans notre environnement de production pour 1 CPU dans nos environnements de test. Le coefficient de 0.5, 500 utilisateurs dans notre environnement de production équivaut à 250 utilisateurs dans nos environnements de test.

Planification des tests

Pour que notre application soit viable, il est impératif que celle-ci soit performante en environnement de production, il faut donc effectuer des tests de performance et de charge pour analyser et évaluer la résilience de notre solution.

Dû à notre choix d'architecture (monolithique) notre application est assez volumineuse, il nous faut donc optimiser le gain d'espace.