PrestaShop

PARISTESTCONF

Reconstruire sa stratégie de tests automatisés

ParisTestConf
26 novembre 2020

Sommaire

REX : comment la QA @ PrestaShop a repensé ses campagnes de tests automatisés de zéro

Simon Garny

@SimonGrn

QA Manager @ PrestaShop

Ancien développeur

Automaticien de coeur

Aime les chats

PrestaShop

Framework ECommerce Open Source depuis 2008 !

Stack classique: PHP, MySQL

Outil très complexe, beaucoup de legacy, en cours de refonte technique: peu d'ajout de fonctionnalités (= beaucoup de non-régression)

Pas de SaaS: on premise

Société: contributrice principale du framework ecommerce

QA: 5 testeurs manuels / 2 testeurs automaticiens

Quelques précisions

Campagne fonctionnelle: tests bas niveau d'UI, cas nominaux, tests simples

E2E: parcours utilisateurs "complexes" utilisant différentes fonctionnalités

Précédente stack (~Q1 2019)

Précédente stack

Une grosse campagne Selenium/Webdriver/Javascript

Subset pour des tests Sanity à chaque Pull Request

Monolithique, pas de séparation des responsabilités

Headless peu fiable, erreurs aléatoires

Scope de la campagne trop large: fonctionnel + E2E

Reporting difficilement utilisable (fichier HTML stocké sur GCP et non utilisé)

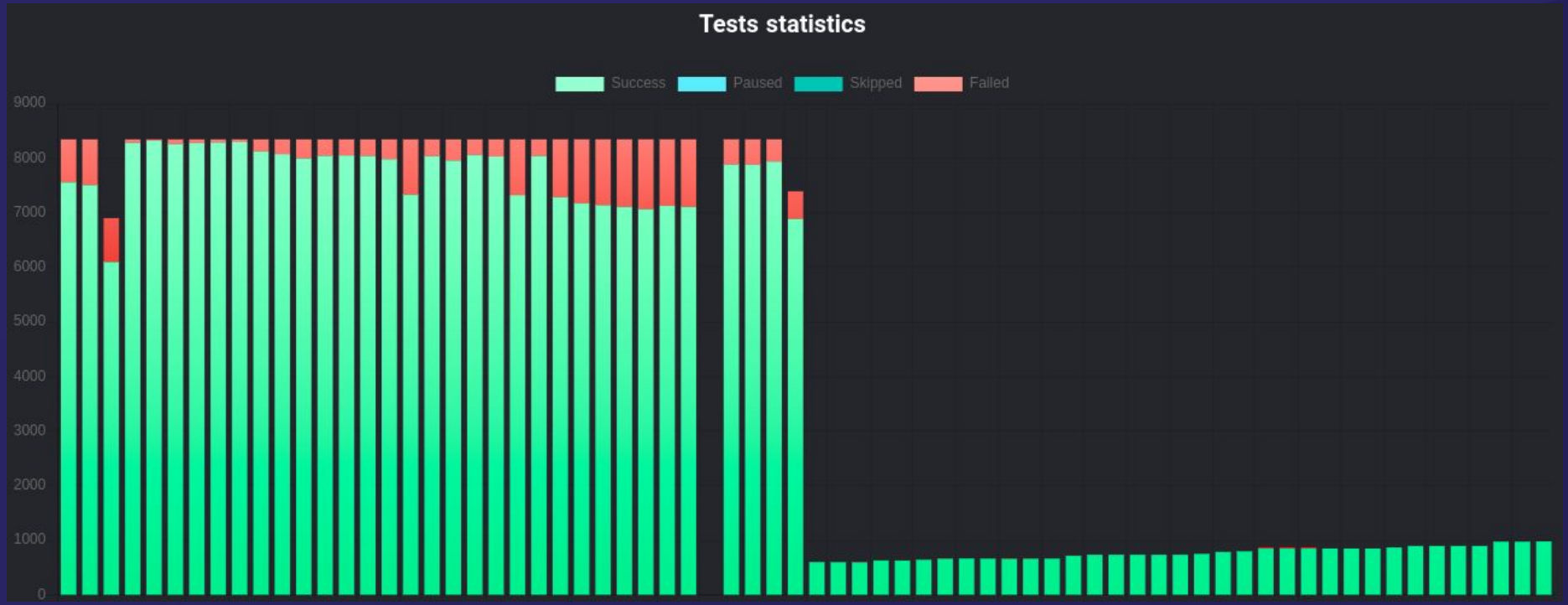
Pas de documentation ⚠️

```
moveFile(downloadsFolderPath, filename, destinationFolder) {
  exec(' mv ' + downloadsFolderPath + filename.split('.')[0] + '.' + filename.split('.')[1] + ' ' + destinationFolder,
    (error) => {
      if (error !== null) {
        console.log(`exec error: ${error}`);
      }
    });
  return this.client
    .pause(3000)
    .refresh();
}
```

```
checkTotalPrice(selector, option :string = 'percent') {
  return this.client
    .pause(2000)
    .then(() => this.client.getText(selector))
    .then( onfulfilled: (code) => {
      if (option === 'amount') {
        expect(code.split('€')[1]).to.be.equal(((tab["totalProducts"].split('€')[1] * 0.5) - 24).toPrecision( precision: 4).toString());
      } else {
        expect(code.split('€')[1]).to.be.equal(((tab["totalProducts"].split('€')[1] * 0.5) * 0.5).toPrecision( precision: 4).toString());
      }
    });
}
```

```
test( title: 'should show the customer form', fn: () => {
  return promise
    .then(() => client.scrollWaitForExistAndClick(AccessPageFO.personal_info, margin: 150, pause: 2000))
    .then(() => client.waitAndSetValue(accountPage.signin_email_input, value: "pub@prestashop.com"))
    .then(() => client.waitAndSetValue(accountPage.signin_password_input, value: "123456789"))
    .then(() => client.waitForExistAndClick(AccessPageFO.login_button))
    .then(() => client.isVisible(accountPage.customer_form))
    .then(() => expect(global.isVisible).to.be.true);
})
```

```
checkAutoUpgrade() {
  fs.readFile( folderPath: rcTarget + 'admin-dev/autoupgrade/tmp/log.txt', fileName: 'utf8', pause: (err, content) => {
    global.upgradeError = content.indexOf("upgradeDbError");
  });
  return this.client
    .pause(2000)
    .then(() => {
      expect(global.upgradeError, message: "Upgrade process done, but some warnings/errors have been found").to.equal( value: -1)
    });
```


Tests statistics
Success
Paused
Skipped
Failed
9000
8000
7000
6000
5000
4000
3000
2000
1000
0

La réflexion (~Q2 2019)

Choix de l’automate

Benchmark de différentes solutions

Cypress, Puppeteer, Selenium

- Totale liberté (pas de vendor locking)
- Open Source
- Headless obligatoire
- Réutilisation des outils connus par l’équipe (Mocha, Chai)
- Reporting adaptable et complet

Choix de l’architecture logicielle

Utiliser un modèle efficace dès le début

POC sur un Design Pattern connu et éprouvé: Page Object Pattern (popularisé par Martin Fowler)

Possibilité d’étendre le système et de l’ouvrir à la communauté (car demandé)

Meilleure transparence (OSS), meilleure documentation

Choix des campagnes

Découpage des scopes

3 niveaux: campagne fonctionnelle (au plus bas niveau possible), campagne Sanity, tests End to End

Couvrir au maximum les cas nominaux simples car:

- Logiciel extrêmement complexe (beaucoup de legacy)
- Logiciel en cours de refonte avec peu d'ajout de fonctionnalités, beaucoup de corrections
- Peu de spécifications disponibles (donc difficile de couvrir des cas tordus)

Une campagne Sanity scopée sur les fonctionnalités critiques, déroulée régulièrement

Une campagne End to End pour les parcours utilisateur critiques

Exécutions & reporting

Exécution de la campagne toutes les nuits sur les branches actives

Exécution des Sanity sur chaque PR sur Github

Meilleur reporting (fiabilité, disponibilité)

Stack actuelle (~Q4 2019)

Stack actuelle

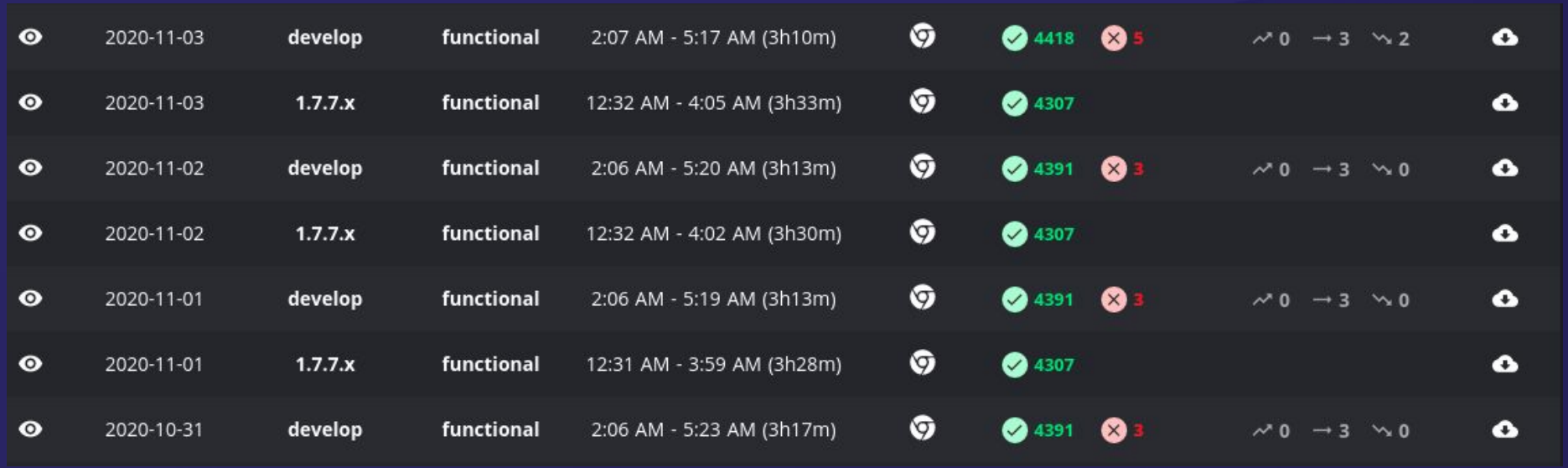
Changement d'automate: Puppeteer puis Playwright (headless natif, fonctionnement ISO en CI / Docker / local)

Documenté et basé sur une campagne écrite dans un outil spécialisé (TestRail)

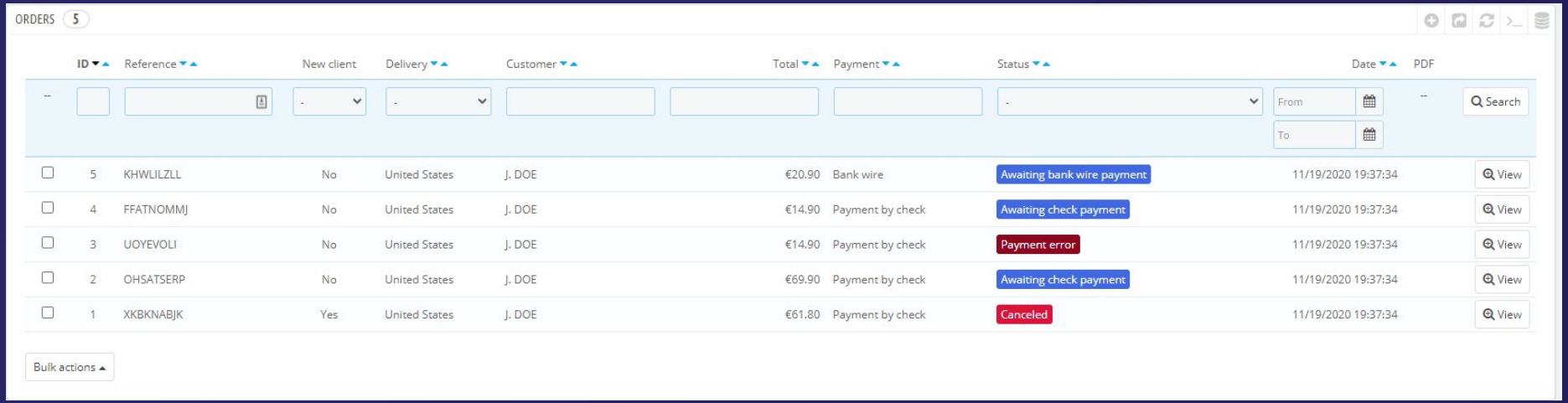
Un Design Pattern éprouvé, documenté, permettant d'écrire un outil maintenable (POP)

Des campagnes correctement scopées

Reporting intégré dans un outil dédié développé pour l'occasion (https://nightly.prestashop.com)


2020-11-03 develop functional 2:07 AM - 5:17 AM (3h10m) 4418 5 0 3 2
2020-11-03 1.7.7.x functional 12:32 AM - 4:05 AM (3h33m) 4307
2020-11-02 develop functional 2:06 AM - 5:20 AM (3h13m) 4391 3 0 3 0
2020-11-02 1.7.7.x functional 12:32 AM - 4:02 AM (3h30m) 4307
2020-11-01 develop functional 2:06 AM - 5:19 AM (3h13m) 4391 3 0 3 0
2020-11-01 1.7.7.x functional 12:31 AM - 3:59 AM (3h28m) 4307
2020-10-31 develop functional 2:06 AM - 5:23 AM (3h17m) 4391 3 0 3 0

La page Orders (commandes)

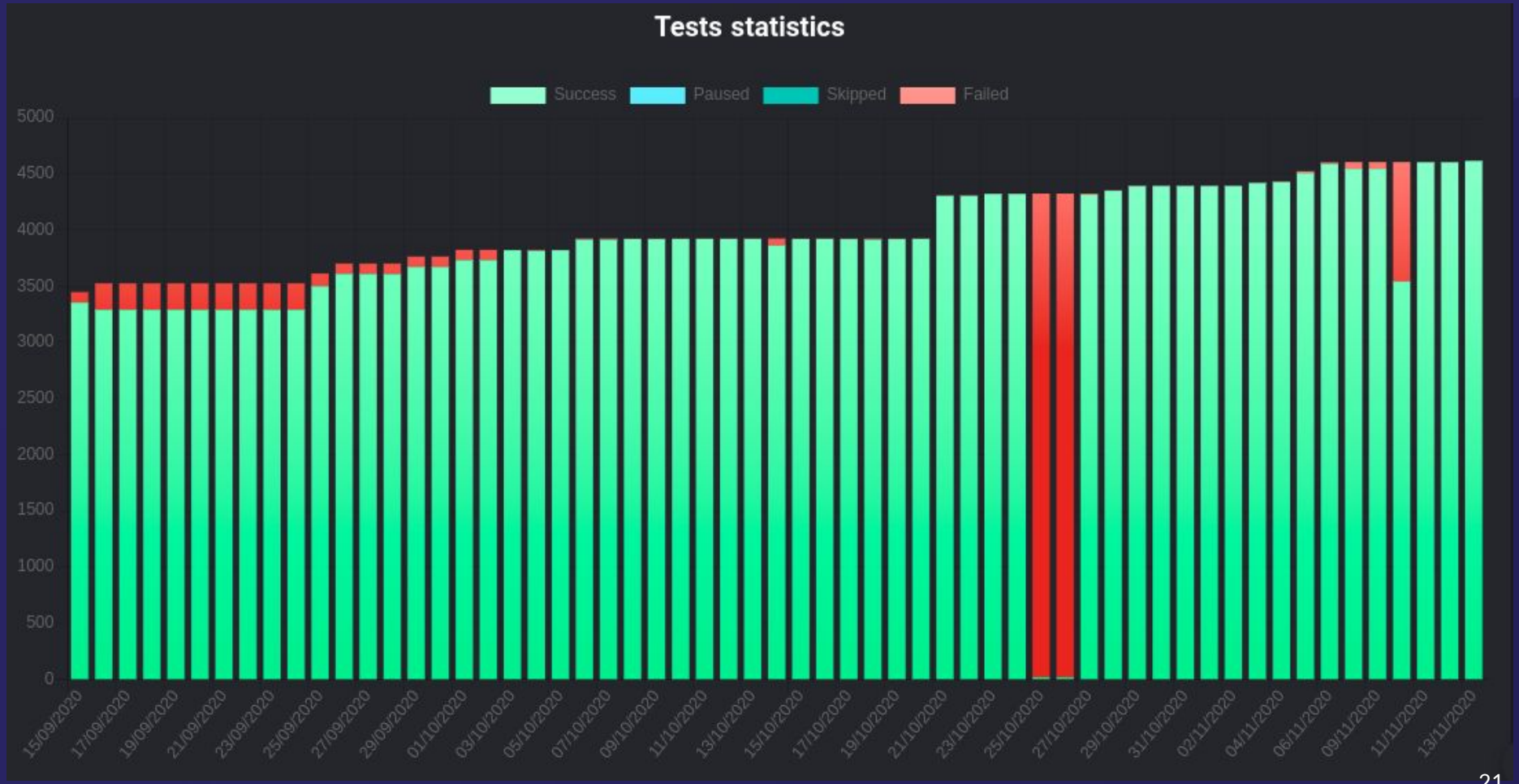


```
it('should reset all filters', async function () {
  const numberOfOrders = await ordersPage.resetAndGetNumberOfLines(page);
  await expect(numberOfOrders).to.be.above(0);
});

it('should filter order by customer name', async function () {
  await ordersPage.filterOrders(
    page,
    filterType: 'input',
    filterBy: 'customer',
    DefaultAccount.lastName,
  );

  const numberOfOrders = await ordersPage.getNumberOfElementInGrid(page);
  await expect(numberOfOrders).to.be.at.least(1);
});
```

```
/**
 * Filter Orders
 * @param page
 * @param filterType
 * @param filterBy
 * @param value
 * @return {Promise<void>}
 */
async filterOrders(page, filterType, filterBy, value : string = '') {
  switch (filterType) {
    case 'input':
      await this.setValue(page, this.filterColumn(filterBy), value.toString());
      break;
    case 'select':
      await this.selectByVisibleText(page, this.filterColumn(filterBy), value);
      break;
    default:
    // Do nothing
  }
  // click on search
  await this.clickAndWaitForNavigation(page, this.filterSearchButton);
}
```


Tests statistics
Success
Paused
Skipped
Failed
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
15/09/2020
17/09/2020
19/09/2020
21/09/2020
23/09/2020
25/09/2020
27/09/2020
29/09/2020
01/10/2020
03/10/2020
05/10/2020
07/10/2020
09/10/2020
11/10/2020
13/10/2020
15/10/2020
17/10/2020
19/10/2020
21/10/2020
23/10/2020
25/10/2020
27/10/2020
29/10/2020
31/10/2020
02/11/2020
04/11/2020
06/11/2020
09/11/2020
11/11/2020
13/11/2020

Le travail des 3 amigos

Les devs travaillent sur la qualité

- Ajout de tests unitaires (PHP Unit)
- Ajout de tests d'intégration (Behat)

Le produit améliore les spécifications

- Rédaction de spécifications
- Tests des spécifications
- Travail avec les utilisateurs finaux

Qu’avons-nous prévu pour la suite ?

A venir

Framework en amélioration continue: refactorisations régulières

Création d'une librairie (esprit OSS)

Evangélisation en interne (process & automatisation) dans les autres pôles

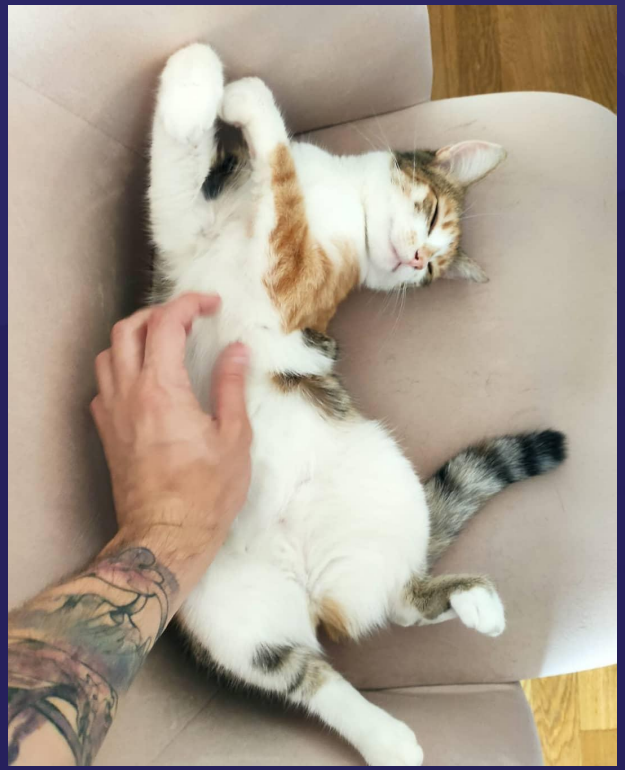
Tests de performance

Tests de webservice

Multi browsers

PrestaShop

PARISTESTCONF



Des questions ? Des remarques ?