formation.tech

Webinaire Managing a web project

Romain Bohdanowicz
Twitter : @bioub - https://github.com/bioub
https://formation.tech/

formation.tech

Introduction

Introduction - Formateur

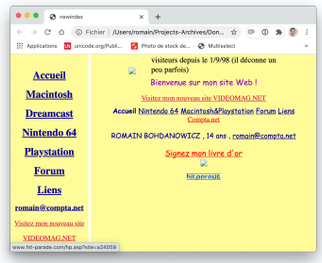
- Romain Bohdanowicz
  Ingénieur EFREI 2008, spécialité en Ingénierie Logicielle

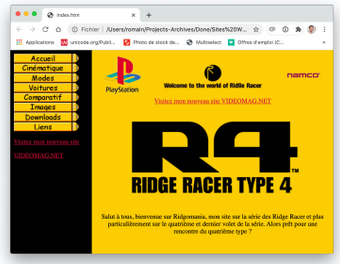
- Expérience
  Formateur/Développeur Freelance depuis 2006
  Près de 2000 jours de formation animées

- Langages
  Expert : HTML / CSS / JavaScript / TypeScript / PHP / Java
  Notions : C / C++ / Objective-C / C# / Python / Bash / Batch

- Certifications
  PHP / Zend Framework / Node.js

- A propos
  Premier site web à 12 ans (HTML/JS/PHP)
  Triathlète du dimanche





Introduction - Horaires

- Matin
  - 9h - 10h
  - 10h15 - 11h15
  - 11h30 - 12h30
- Après-midi
  - 13h45 - 14h45
  - 15h - 16h
  - 16h15 - 17h15
- Questionnaire de satisfaction à remplir en fin de formation : https://stagiaire.formation.tech/

Introduction - formation.tech

- Organisme de formation depuis 2016
- Référencé DataDock
- Certifié Qualiopi
- 15 formations au catalogue
- Une dizaine de formateurs indépendants
- Formations en français ou anglais
- https://formation.tech/

Qualiopi
processus certifié

DATADOCKÉ

Introduction - WeAreDevs

- Studio de développement créé en 2017
- 1 salarié développeur sénior
- Principales références
  - Cinexpert / Adeum
  - Sponsorise.me
  - Intel
  - Staytuned
  - STMicroelectronics
- https://wearedevs.fr/

Introduction - Et vous ?

- Rôle chez STMicroelectronics ?
- Intérêt / objectif de cette formation ?

formation.tech

Les étapes d'un projet web

Les étapes d'un projet web

- Conception
  - Cahier des charges fonctionnel
  - Mockup / Wireframing
  - Maquettes graphiques
  - Cahier des charges technique
- Développement
  - Langages
  - Bibliothèques / Frameworks
  - Outils
- Recette
  - Tests
  - Légal
  - CI / CD
- Mise en production et suivi
  - Hébergement
  - Déploiement
  - Domaine / DNS
  - Cache
  - Monitoring
  - Référencement
  - Accessibilité

formation.tech

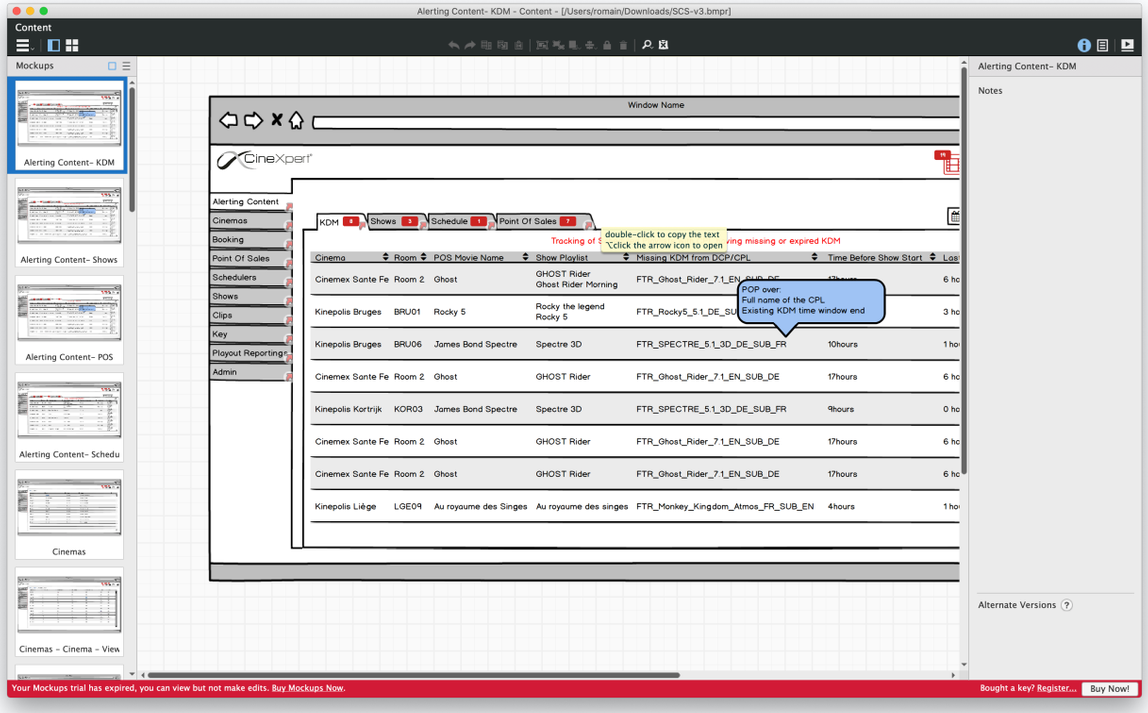
Conception

Conception - Cahier des charges fonctionnel

- Après recueil du besoin
- Rédaction sous Word ou autre
- Identifier les acteurs (utilisateurs, admin...)
- Penser "pages", décrire le parcours utilisateur
- Quelles plateformes cibles ? Web ? App ? Desktop ? Mobile ? IoT ?
- Eventuellement RGPD
- Accessibilité ?

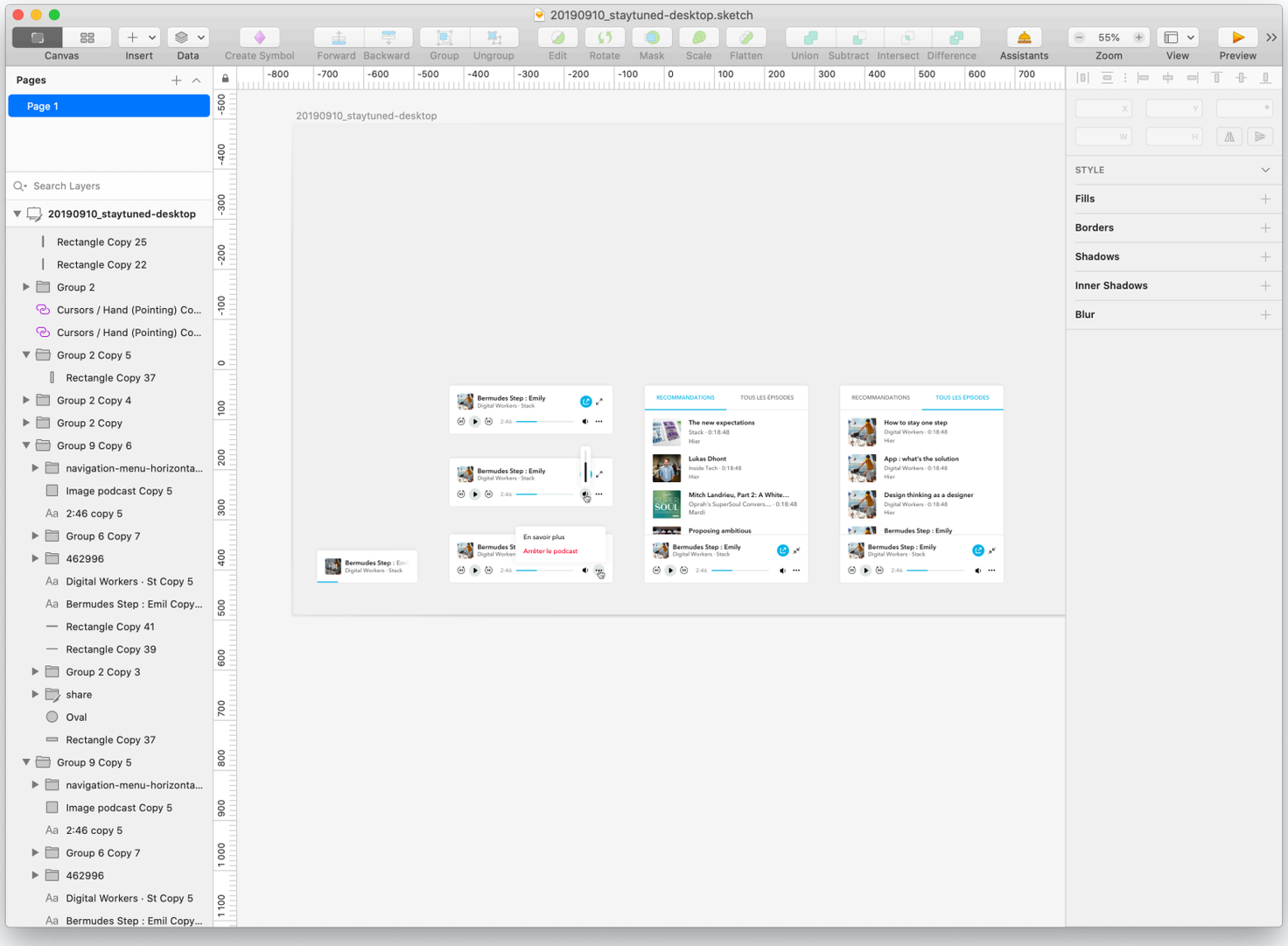
Conception - Wireframing

- Dessiner les pages
  - UX : eXpérience Utilisateur (UX Designer)
  - déposer les éléments d'UI sur les pages
  - fixed design / fluid design
  - anticiper version desktop / mobile -> responsive design
  - développement Mobile first
  - test du parcours utilisateur
- Outils
  - Balsamiq
  - Marvel
  - Invision
  - Un papier et un crayon



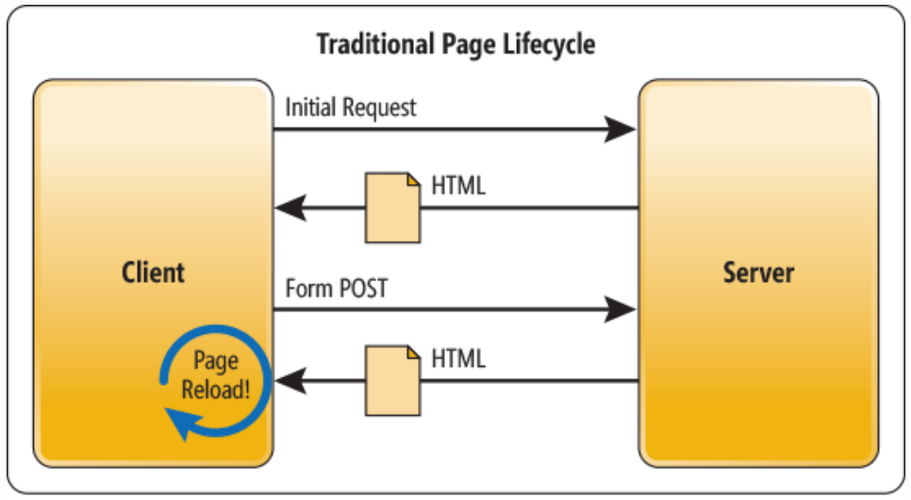
Conception - Maquettes

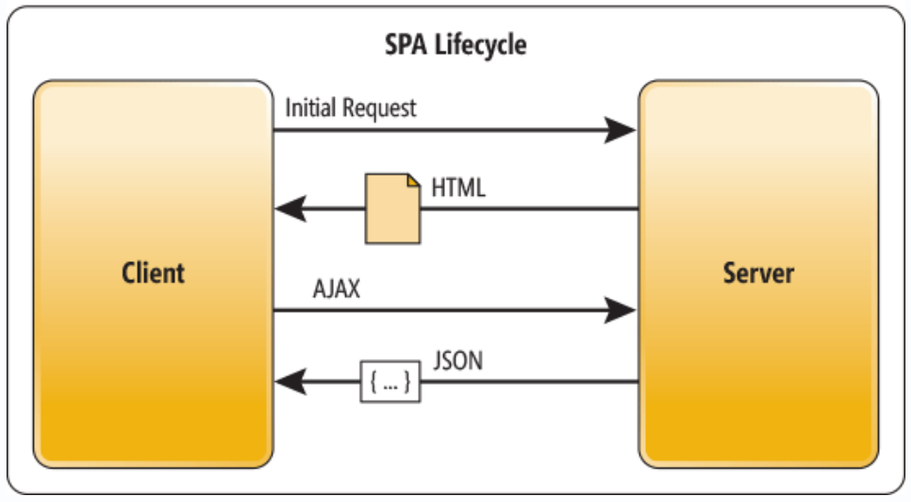
- Maquettes graphique
  - Outils historiques : Photoshop / Illustrator / Fireworks (abandonné)
  - Modernes : Sketch / Figma / Adobe xd



Conception - Cahier des charges technique

- Un outil existe t'il déjà ?
  - CMS (Content Management System) : Wordpress, Drupal...
  - Générateur de site statique : Jekyll, Gatsby... (format Markdown)
  - Logiciel / plateforme :
    - e-commerce : Prestashop / Magento / Shopify
    - Site web : Wix / Webflow
    - ...
- Développement spécifique
  - Webapp "traditionnelle"
  - Single Page Application





Conception - Cahier des charges technique

- Plateformes
  - Web desktop
  - Web mobile
  - App desktop
  - App mobile
  - Autres (TV, Montres connectées...)
- Langages et protocoles
  - Communication : HTTP, Services Web
  - Front-end : HTML, CSS, JavaScript (WebAssembly)
  - Back-end :
    - Historiques : PHP, Python, Ruby...
    - Entreprises : Java, C#...
    - Récents : JavaScript, Go...

Conception - Cahier des charges technique

- Bases de données
  - Relationnelles : MySQL, PostgreSQL...
  - NoSQL : Redis, MongoDB, Cassandra...
- Choix ?
  - Modèle entité-association (Merise, UML)
    - beaucoup de relations -> base relationnelles (= intégrité)
    - beaucoup d'imbrication -> orienté document MongoDB / CouchDB
    - beaucoup de données géographiques -> PostGIS / MongoDB...
    - cache ? perf ? -> Memcached
    - recherche ? log ? -> ElasticSearch
    - embedded -> Sqlite, PouchDB, LevelDB
  - Environnements de plus en plus hétérogènes
- Stacks techniques
  - historiques : LAMP (Linux Apache MySQL PHP), MEAN (Mongo Express AngularJS Node)...
  - hétérogènes : https://stackshare.io/stacks

Conception - Cahier des charges technique

- Hébergement
  - Bare metal / On premise
  - Mutualisé
  - Virtualisé
  - Platform as a service
  - Conteneurisé
  - Serverless / Function as a service
- Le choix de l'hébergement peut influer sur les choix technologiques
  - Bare Metal / Mutualisé / Virtualisé => maintenance des apps
  - Platform as a service / Conteneurisé => facilite l'environnement hétérogène
  - Serverless => Node.js / Python plus performants

formation.tech

Développement

Développement - Docs

- Developer Roadmaps
  - https://github.com/kamranahmedse/developer-roadmap
  - https://roadmap.sh
- Project Guidelines / Style Guides
  - https://github.com/elsewhencode/project-guidelines
  - https://google.github.io/styleguide/
- Livres
  - https://github.com/EbookFoundation/free-programming-books
  - https://github.com/mtdvio/every-programmer-should-know
  - https://github.com/kamranahmedse/design-patterns-for-humans

Développement - Docs

- JavaScript
  - https://github.com/mbeaudru/modern-js-cheatsheet
  - https://github.com/goldbergyoni/nodebestpractices
  - https://github.com/ryanmcdermott/clean-code-javascript
- Front-end
  - https://github.com/krausest/js-framework-benchmark
  - https://github.com/andrew--r/frontend-case-studies
  - https://github.com/thedaviddias/Front-End-Checklist
  - https://github.com/FrontendMasters/front-end-handbook-2019
  - https://github.com/moklick/frontend-stuff

Développement - Docs

- Les Awesome Lists de Github
  - https://github.com/topics/awesome
  - https://github.com/sindresorhus/awesome
  - https://github.com/vinta/awesome-python
  - https://github.com/avelino/awesome-go
  - https://github.com/jaywcjlove/awesome-mac
  - https://github.com/sindresorhus/awesome-nodejs

Développement - IDE

- Les IDEs
  - Vim / Emacs
  - Eclipse / Netbeans
  - Visual Studio / Xcode / IntelliJ et dérivés
  - Sublime Text / Atom
  - Visual Studio Code / Theia
- Le fichier .editorconfig -> https://editorconfig.org

Développement - Front-end

- Langages et HTTP
  - HTTP : communication avec le serveur
  - HTML : description du contenu
  - CSS : mise en forme
  - JavaScript : comportement dynamique
- Bibliothèques
  - Composants graphiques (UI Kit) : Bootstrap, Material, Primefaces, Tailwind, Bulma...
  - Frameworks :
    - 1ère génération : Prototype.js, jQuery, Dojo, Mootools
    - 2nde génération : Backbone, AngularJS, Ember.js, Knockout
    - 3e génération : React, Vue.js, Angular
  - Web components : Polymer, Stencil

Développement - Front-end

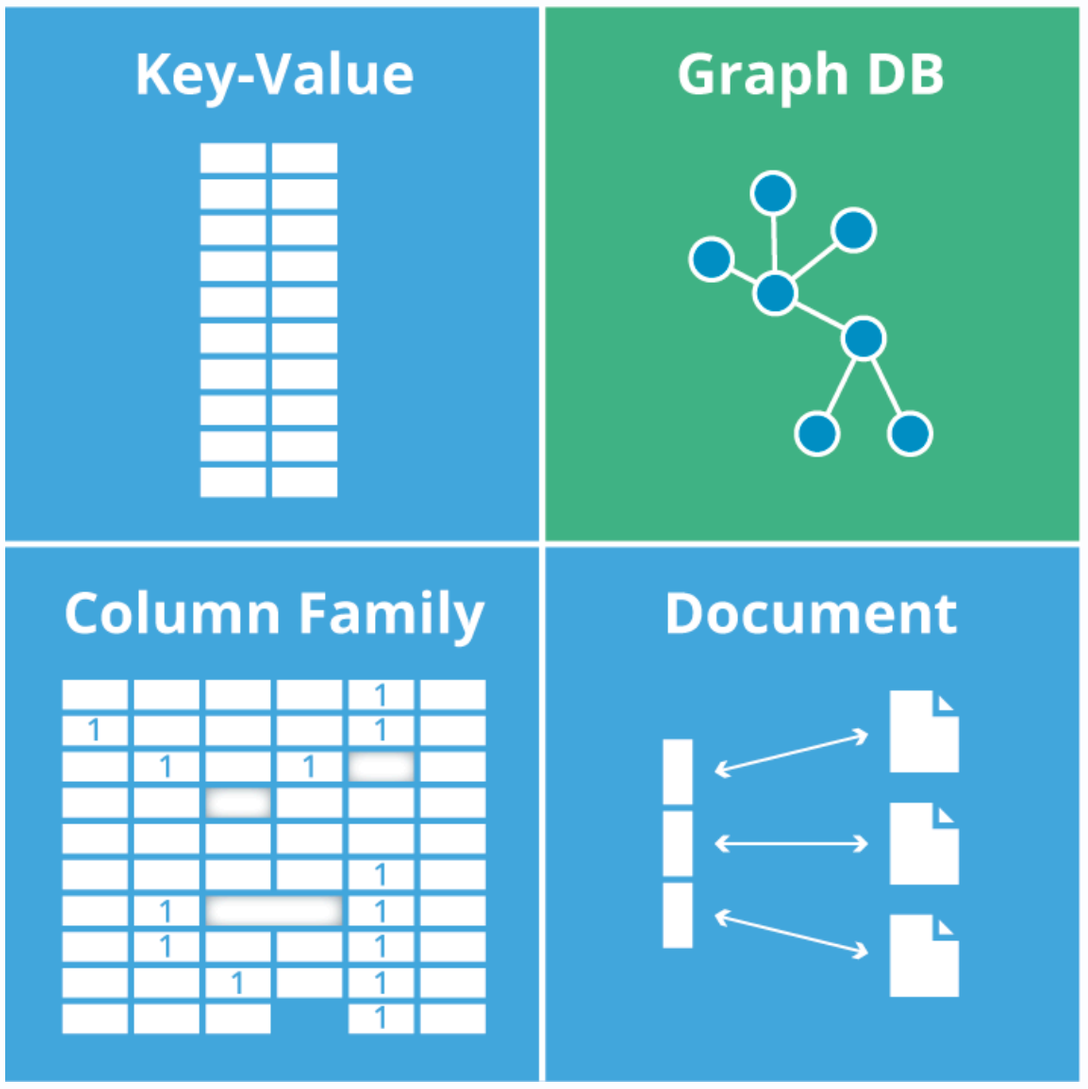
- Outils
  - Package managers : npm, Yarn
  - Préprocesseurs CSS / Postprocesseurs : SASS, PostCSS
  - Bundler : Webpack, Rollup
  - Task runners : scripts npm, Gulp
  - Transpilers : TypeScript, Babel

Développement - Back-end

- Choix du langage
  - Compétences existantes
  - Bibliothèque existante
  - Performances
  - Universalité
  - Facilité d'apprentissage
  - Communauté

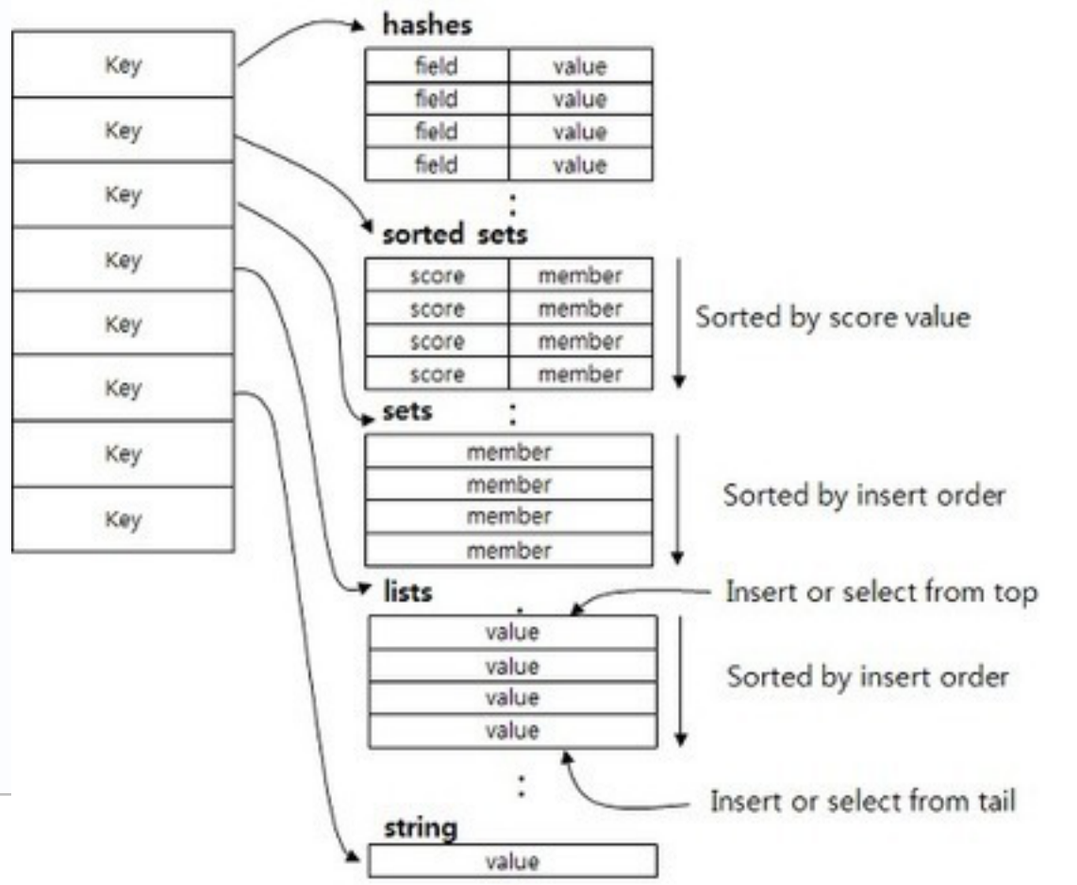
Développement - Back-end

- Base de données relationnelles
- Base de données NoSQL
  - Key-Value : Redis, Memcached
  - Column : Cassandra, HBase
  - Graph : Neo4j
  - Document : CouchDB, MongoDB



NoSQL - Clé / valeur

- Exemple Redis




```
const redis = require("redis");
const client = redis.createClient();

client.on("error", function(error) {
  console.error(error);
});

client.set("key", "value", redis.print);
client.get("key", redis.print);
```

NoSQL - Orienté Colonne

- Exemple Cassandra

```
const cassandra = require('cassandra-driver');

const client = new cassandra.Client({
  contactPoints: ['h1', 'h2'],
  localDataCenter: 'datacenter1',
  keyspace: 'ks1'
});

const query = 'SELECT name, email FROM users WHERE key = ?';

client.execute(query, [ 'someone' ])
  .then(result => console.log('User with email %s', result.rows[0].email));
```

	A	B	C	D	E
1	foo	bar	hello		
2		Tom			
3			java	scala	cobol

Organisation d'une table dans une BDD relationnelle

1	A foo	B bar	C hello
2	B Tom		
3	C java	D scala	E cobol

Organisation d'une table dans une BDD orientée colonnes

NoSQL - Orienté Document

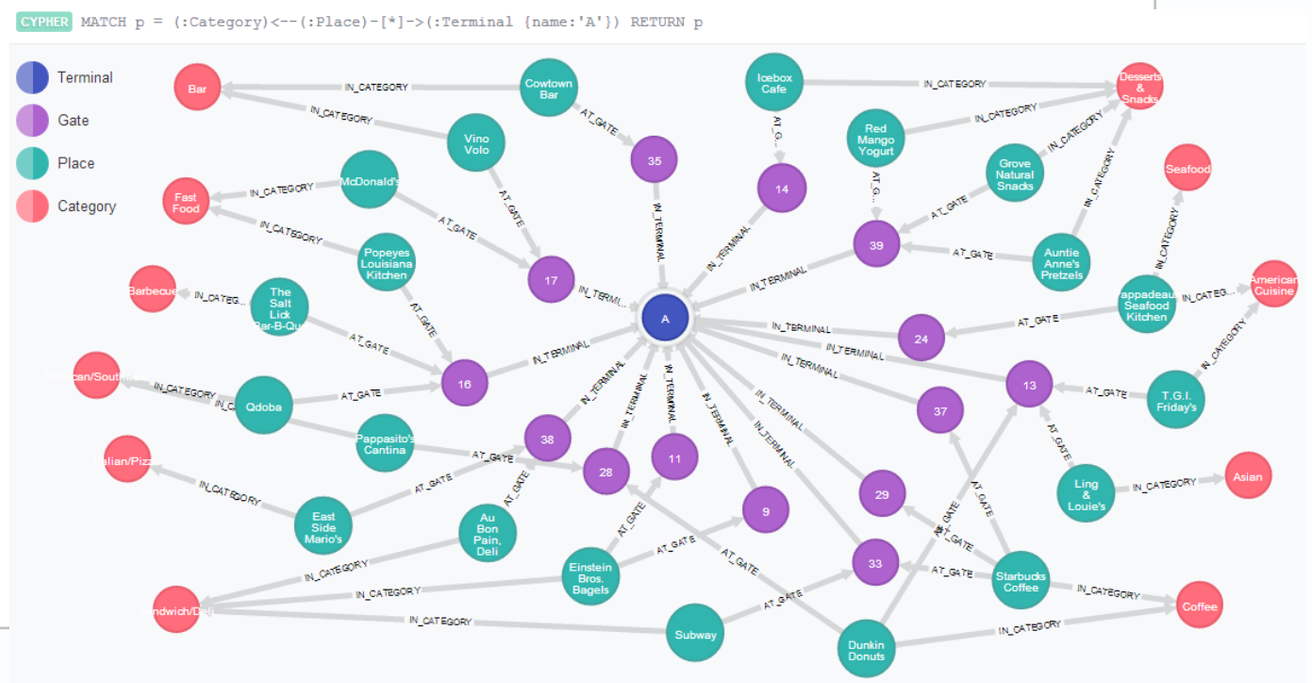
- Exemple CouchDB

```
const cradle = require('cradle');
const db = new(cradle.Connection)().database('starwars');

db.get('vader', function (err, doc) {
    doc.name; // 'Darth Vader'
    assert.equal(doc.force, 'dark');
});

db.save('skywalker', {
    force: 'light',
    name: 'Luke Skywalker'
}, function (err, res) {
    if (err) {
        // Handle error
    } else {
        // Handle success
    }
});
```

NoSQL - Orienté Graphe

- Exemple neo4j

```
const neo4j = require('neo4j-driver')

const driver = neo4j.driver(uri, neo4j.auth.basic(user, password))
const session = driver.session()
const personName = 'Alice'

try {
  const result = await session.run(
    'CREATE (a:Person {name: $name}) RETURN a',
    { name: personName }
  )

  const singleRecord = result.records[0]
  const node = singleRecord.get(0)

  console.log(node.properties.name)
} finally {
  await session.close()
}

// on application exit:
await driver.close()
```



Développement - Back-end

- Stateful vs Stateless
- Services Web : SOAP, REST, XML-RPC, JSON-RPC, GraphQL (React-Admin)
- Authentification : Tokens, JWT, OAuth
- Authorization : Roles vs ACL
- File de messages : RabbitMQ

Développement - Git

- Bonnes pratiques
  - Feature branch
  - Choisir / écrire un workflow
  - Utiliser les hooks
  - Conventionnal Commit -> https://www.conventionalcommits.org/en/v1.0.0/
  - Intégration / Déploiement continu
- Les plateformes
  - Github
  - Gitlab
  - Bitbucket

formation.tech

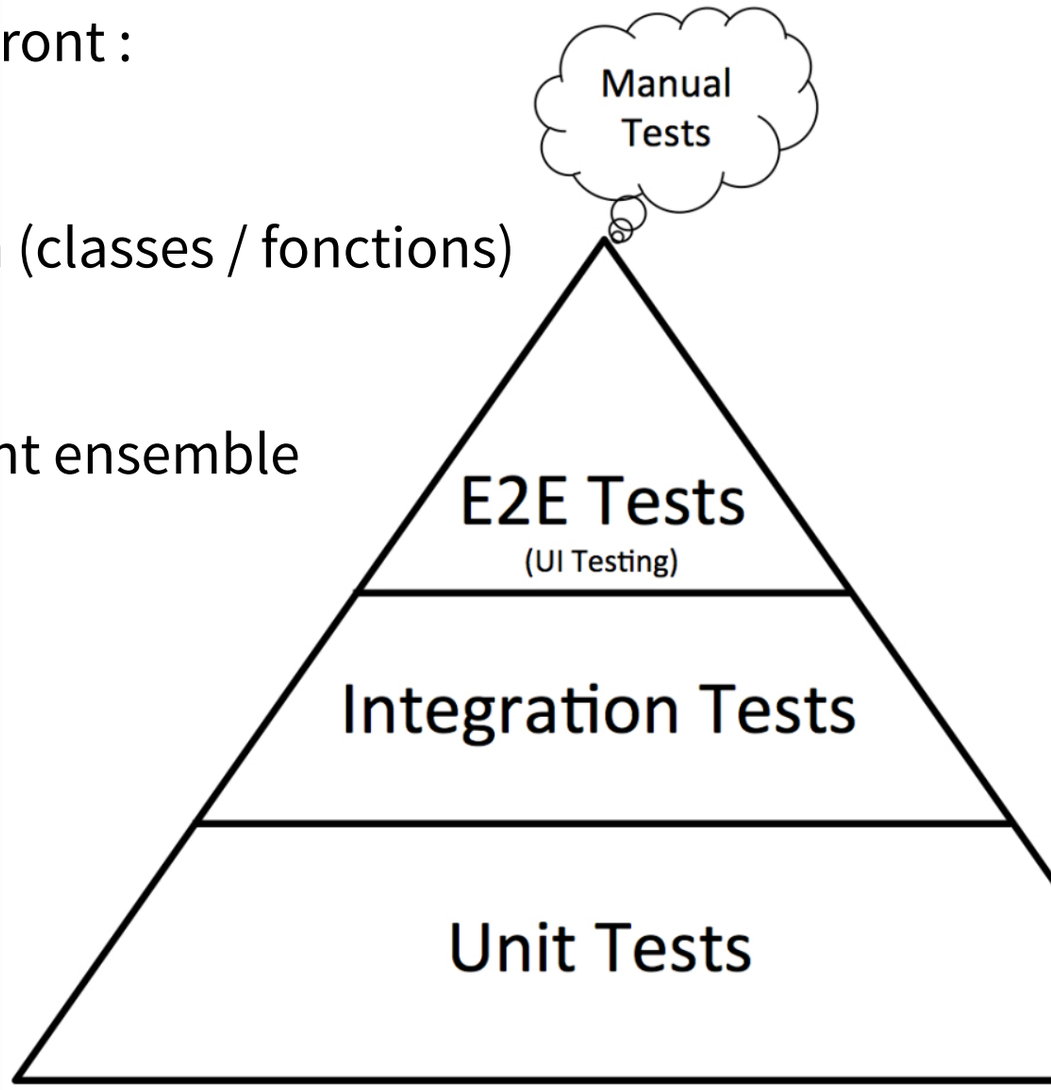
Recette

Recette - Analyse statique

- Convention de code
  - Google https://google.github.io/styleguide/
  - Airbnb https://github.com/airbnb/javascript
  - Standard https://standardjs.com/
- Linters
  - Eslint
  - Sonarqube
  - Stylelint
  - Htmllint
- Code formatter
  - Eslint
  - Prettier

Recette - Tests Automatisés

- Avec les tests automatisés, les scénarios de tests sont codés et peuvent être rejoués rapidement plus régulièrement.
- 3 types de tests automatisés au niveau code côté Front :
  - Test unitaire
    Permets de tester les briques d'une application (classes / fonctions)
  - Test d'intégration
    Teste que les briques fonctionnent correctement ensemble
  - Test End-to-End (E2E)
    Vérifie l'application dans le client



Recette - Test Unitaires

- Voici à quoi ressemble un test unitaire simple écrit avec la bibliothèque Mocha

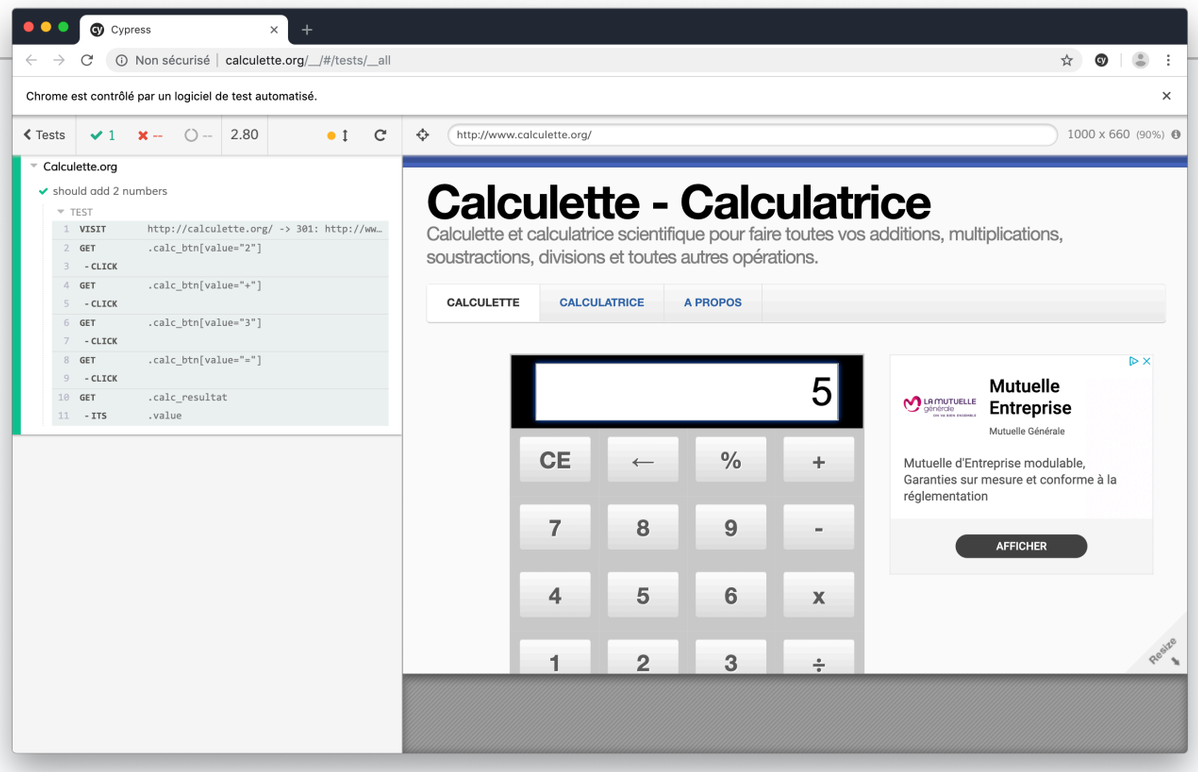
```
const assert = require('assert');
const { ajouter } = require('../src/calculette');

describe('Test calculette', () => {
  it('2 + 3 devraient faire 5', () => {
    assert.equal(5, ajouter(2, 3));
  });
});
```

- Le test unitaire ne couvre le code que d'une seule fonction :
  - il est très rapide à exécuter
  - peu couteux à écrire
  - permet d'identifier rapidement l'origine du problème (la fonction testée)

Recette - Test E2E

- Voici à quoi ressemble un test E2E écrit avec Cypress

```
describe('Calculette.org', () => {
  it('should add 2 numbers', () => {
    cy.visit('http://calculette.org/')

    cy.get('.calc_btn[value="2"]').click();
    cy.get('.calc_btn[value="+"]').click();
    cy.get('.calc_btn[value="3"]').click();
    cy.get('.calc_btn[value="="]').click();

    cy.get('.calc_resultat').its('value').should('be', '5');
  });
});
```



Recette - Tests Multinavigateurs

- Des plates-formes comme BrowserStack ou SauceLabs permettent de :
  - lancer des navigateurs depuis leur site pour tester manuellement
  - lancer des tests écrits avec des frameworks E2E
- BrowserStack supporte plus de 2000 navigateurs
- Ces plates-formes sont payantes (à partir de $29 par mois)

Recette - Licences

- Copyleft : oblige les projets qui dépendent de la licence a être également open-source
  - Exemple de licence copyleft : GNU GPL, Cecill
  - Exemple de licence non-copyleft : MIT, BSD, Apache
- Pas de licence = pas utilisable
- https://choosealicense.com
- https://opensource.guide/legal/

Recette - CI / CD

- Intégration / Déploiement Continu
  - Jenkins
  - Travis
  - Gitlab CI
  - Github Actions

formation.tech

Mise en production

Mise en production - Déploiement

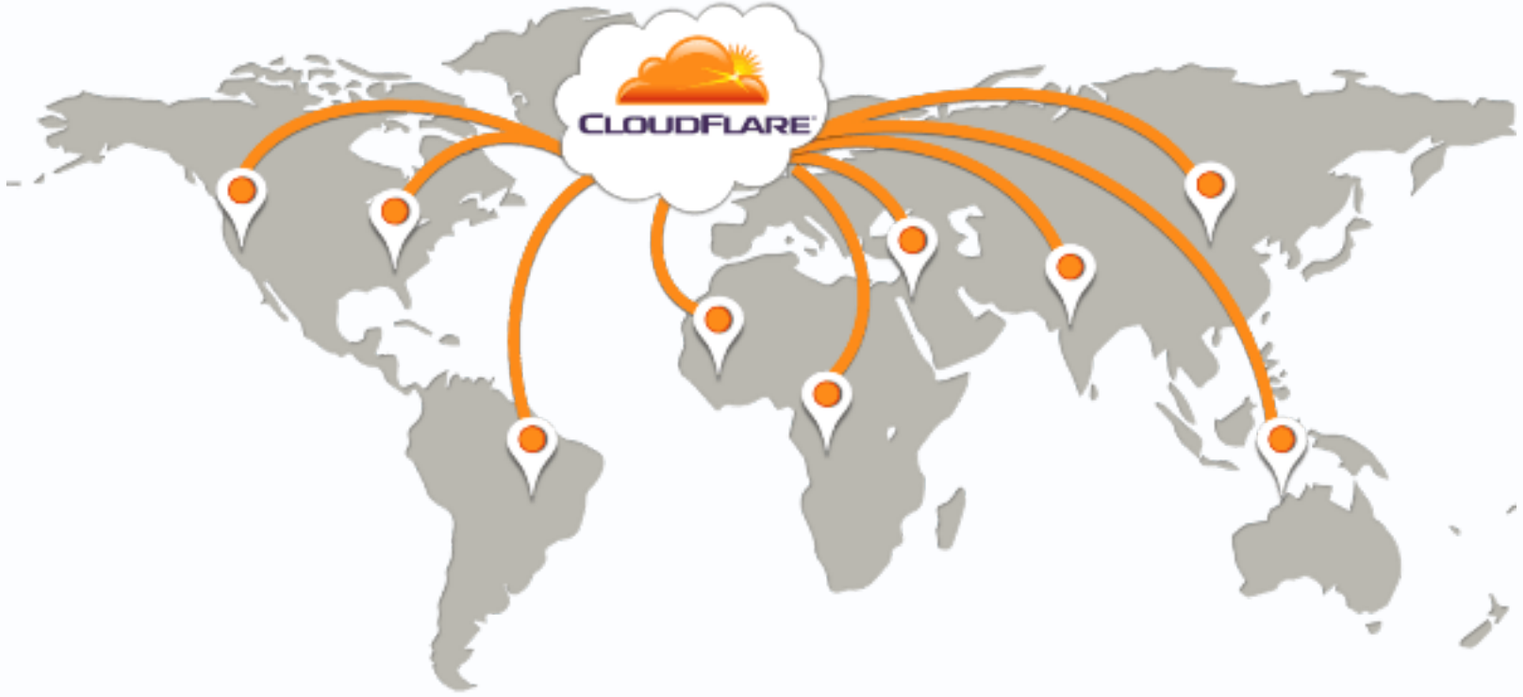
- Outils de transferts : ftp, scp, git
- Automatisation : Scripts Shell, Capistrano

Mise en production - Domaine

- Registry : grossiste en domaines pour une extension donnée
- Registrar : revendeur de nom de domaine
- Whois : programme permettant de connaitre le propriétaire d'un domaine
- Entrée DNS : lien entre le domaine et le serveur
- Serveur DNS : annuaire

Mise en production - Cache

- Cache client : navigateur
- Cache serveur : Varnish, Cloudflare
- Content Delivery Network : Akamai, Cloudflare



Suivi - Monitoring

- Gestion des logs : Elastic Logstash
- Monitoring applicatif : New Relic