

# Génie Logiciel

## Technologies du Web et Images Numériques (TWIN2)

Bobet Goualo Victorien

Enseignant-Chercheur à l'Ecole Supérieure Africaine des TIC

Ingénieur de conception en Sciences et Technologies de l'Information et de la Communication (STIC) de l'INP-HB

Executive Master en Big Data & Data Science de ISAE-ISM Paris

Certifié Oracle, Centrale supélec, Orange Digital Center (ODC)

[victorien.bobet@esatic.edu.ci](mailto:victorien.bobet@esatic.edu.ci)

# Prérequis au cours

- Logique mathématique
- L'industrie du logiciel
- Capacité d'écoute
- Qualités rédactionnelles

# Prérequis au cours

- Logique mathématique

Celle-ci vous apporte la rigueur dans la conception de bons logiciels quelque soit le niveau de complexité. En effet cela vous aide à réfléchir, à analyser correctement les cas et à gérer les erreurs dans votre logiciel.

- L'industrie du logiciel

- ✓ La crise du logiciel
- ✓ Les entreprises de technologies;
- ✓ Les éditeurs de logiciels;
- ✓ Les entreprises de services qui revendent les logiciels en les intégrant dans les SI des entreprises.

N.B: Un travail de recherche à effectuer sur l'industrie du logiciel

# Prérequis au cours

- Capacité d'écoute

Savoir écouter le client pour produire un logiciel qui répond le mieux possible aux besoins exprimés.

- Qualités rédactionnelles

Savoir rédiger des rapports (fond et forme) de qualité.

# Prérequis au cours

## Exercice 0

1. Qu'est-ce qu'un système d'information ?
2. Qu'est-ce qu'un système d'exploitation ? En citer deux exemples ?
3. Quelle est la différence entre une base de données et un Data Center ?
4. Citer trois exemples bases de données.
5. Une entreprise vous demande d'élaborer une application web et mobile, qu'est-ce que vous faites ? Autrement quelle est votre démarche ?
6. Qu'est-ce qu'un langage de programmation ? Citer trois exemples.
7. Qu'est-ce qu'un Cloud ? En citer deux exemples.
8. Pourquoi les entreprises hésitent à adopter le Cloud ?
9. Qui sont les gros consommateurs du Cloud ? En expliquer les raisons.

# Prérequis au cours

## Exercice 0 corrigé

1. Qu'est-ce qu'un système d'information ?

**Un système d'information est l'ensemble de toutes les informations qui circulent au sein de l'entreprise et les moyens mis en œuvre pour les gérer.**

2. Qu'est-ce qu'un système d'exploitation ? En citer deux exemples ?

**Un système d'exploitation (OS) est un logiciel de base ou un ensemble de programme, qui permet d'utiliser l'ordinateur de façon optimale et équitable.**

**Ex: Windows, CentOS, Ubuntu, Oracle linux, Mac OS, Suse linux, Red Hat ...**

# Prérequis au cours

## Exercice 0 corrigé

3. Quelle est la différence entre une base de données et un Data Center ?

**Un serveur de base de données est une machine tandis qu'un Data Center est un ensemble de machines (un ensemble de plusieurs serveurs). La base de données est incluse dans le Data Center.**

4. Citer trois exemples bases de données.

**Base de données relationnelles : MySQL, SQL Server, Oracle, PostgreSQL, Maria DB, ...**

**Base de données NoSQL : Mongo DB, Cassandra, Hbase, Neo4j, Cassandra, Oracle NoSQL, ...**

# Prérequis au cours

## Exercice 0 corrigé

5. Une entreprise vous demande d'élaborer une application web et mobile, qu'est-ce que vous faites ? Autrement quelle est votre démarche ?

**Bien écouter le client pour mieux cerner son besoin, les fonctionnalités de l'application qu'il souhaite qu'on mette à sa disposition.**

6. Qu'est-ce qu'un langage de programmation ? Citer trois exemples.

**C'est l'ensemble de caractères, de symboles et de règles syntaxiques utilisés dans l'écriture des programmes.**

**Ex : Python, C, C++, Pascal, Php, Java, R, etc.**



# Prérequis au cours

## Exercice 0 corrigé

7. Qu'est-ce qu'un Cloud ? En citer deux exemples.

C'est un ensemble de services (éventuellement informatiques) hébergés sur des data centers de fournisseurs de services cloud accessible via internet.

Ex : Oracle Cloud, Microsoft Azure, Google Cloud

Attention un service cloud correspond à l'un des services offerts par les fournisseurs (IaaS, SaaS, LaaS, DaaS, PaaS ...)

IaaS : Infrastructure as a Service, DaaS : Data ou Database as a Service,  
PaaS : Plateforme as a Service, SaaS : Software as a Service, LaaS : Licensing as a Service

# Prérequis au cours

## Exercice 0 corrigé

8. Pourquoi les entreprises hésitent à adopter le Cloud ?

Les entreprises craignent l'espionnage puisqu'elles n'auront pas le contrôle absolu sur leurs données (autrement elles ne seront pas les supers admin de leur data).

Les fournisseurs de cloud proposent des services DaaS. Autrement proposent des données au plus offrant comme service. La question qui se pose : quelles sont ces données commercialisées comme services ? (une réflexion à mener)

# Prérequis au cours

## Exercice 0 corrigé

9. Qui sont les gros consommateurs du Cloud ? En expliquer les raisons.

**En majorité, nous avons les PME, Startup. Autrement les entreprises qui ont peu de ressources pour s'offrir des data centers comme les grands groupes. Et même des grands groupes optent pour le cloud lorsque leur cœur de métier n'est pas l'informatique. Ainsi cela leur évite la charge de maintenance d'équipements informatiques qui est gérée directement par le fournisseur de services cloud.**

# Plan du cours

- I. Généralités
- II. Etude de faisabilité et analyse des besoins
- III. Architectural Patterns
- IV. Conception de la solution
- V. Les ateliers de génie logiciel
- VI. Méthodes agiles
- VII. Implémentation logicielle
- VIII. Qualité logicielle
- IX. Gestion de la configuration
- X. Maintenance logicielle
- XI. Gestion de projets logiciel

# I. Généralités

## Logiciel

Un logiciel est un ensemble de séquences d'instructions interprétables par une machine et d'un jeu de données nécessaires à ces opérations.

Un logiciel est l'ensemble des programmes destiné au fonctionnement du matériel et à la résolution de certains problèmes spécifiques.

Un « logiciel » est, selon le vocabulaire officiel de l'informatique, l'« ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données.

# I. Généralités

## Génie logiciel

Le génie logiciel (ou ingénierie des systèmes d'information) représente l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi.

Autrement dit l'art de produire de bons logiciels, au meilleur rapport qualité/prix

# I. Généralités

Pourquoi le génie logiciel en entreprise ?

- Complexité des logiciels
- Indépendance logicielle des entreprises
- Extension des applications (ajout de fonctionnalités)
- Le besoin grandissant d'automatisation

# I. Généralités

Les métiers autour du génie logiciel ?

- Ingénieur logiciel
- Développeur Web & Mobile
- Devops Engineer
- Back end Engineer
- Front end Engineer
- Full Stack Engineer
- Testeur
- ...



# I. Généralités

## Les métiers autour du génie logiciel ?

- Ingénieur logiciel

[https://www.welcometothejungle.com/fr/companies/orange-1/jobs/ingenieur-logiciel-f-h-en-contrat-d-apprentissage\\_courbevoie](https://www.welcometothejungle.com/fr/companies/orange-1/jobs/ingenieur-logiciel-f-h-en-contrat-d-apprentissage_courbevoie)

<https://www.wizbii.com/company/orange/job/ingenieur-logiciel-devops-f-h>

- Devops Developer or Engineer

<https://orange.jobs/jobs/v3/offers/116750?lang=fr#:~:text=votre%20r%C3%B4le,en%20%C5%93uvre%20des%20solutions%20DevOps>

- Back end Developer or Engineer

<https://orange.jobs/jobs/v3/offers/99062?lang=FR>

[https://www.bizao.com/offres\\_d\\_emploi/ingenieur-developpeur-back-end-h-f/](https://www.bizao.com/offres_d_emploi/ingenieur-developpeur-back-end-h-f/)

- Front end Developer or Engineer

- <https://orange.jobs/jobs/v3/offers/107650?lang=fr>

- Full Stack Developer or Engineer

<https://www.freecodecamp.org/news/what-is-a-full-stack-developer-back-end-front-end-full-stack-engineer/#:~:text=What%20is%20Full%20Stack%20Development,and%20infrastructure%20of%20the%20site.>

[https://www.bizao.com/offres\\_d\\_emploi/ingenieur-developpeur-full-stack-h-f/](https://www.bizao.com/offres_d_emploi/ingenieur-developpeur-full-stack-h-f/)

[https://www.bizao.com/offres\\_d\\_emploi/lead-developer-mobile-and-web/](https://www.bizao.com/offres_d_emploi/lead-developer-mobile-and-web/)

<http://emploi.educarriere.ci/offre-16977-developpeur-dapplications-debutants-confirmer.html>

- ...

# I. Généralités

Les certifications appréciées par les entreprises ?

- Java SE, Java EE

[https://education.oracle.com/oracle-certification-path/pFamily\\_48](https://education.oracle.com/oracle-certification-path/pFamily_48)

- Devops (Git, Gitlab, Github, Docker, Kubernetes, Cloud)

<https://about.gitlab.com/handbook/customer-success/professional-services-engineering/gitlab-technical-certifications/>

- MySQL, Oracle, SQL Server, MongoDB

<https://www.oracle.com/fr/corporate/features/oracle-certification.html>

<https://university.mongodb.com/certification>

# II. Etude de faisabilité et analyse des besoins

## Expressions des besoins

Contexte, périmètre du projet

Objectif général, principal, globale

Objectifs secondaires, spécifiques

Contraintes

Existant

Critique de l'existant

Proposition de solutions

## II. Etude de faisabilité et analyse des besoins

### **Cahier de charge**

Un document qui indique clairement les besoins du client (ou de l'entreprise) en termes de fonctionnalités.

En tant que génie logiciel, à vous de voir la faisabilité du logiciel demandé !

### **Cahier de charge (fonctionnel)**

Permet de lister les fonctionnalités de la solution à réaliser. il est rédigé en langage simple et validable par le client.

### **Cahier de charge (Technique)**

Décrit la structure et le comportement. Il est rédigé en langage technique et validable par les développeurs

# II. Etude de faisabilité et analyse des besoins

## Cahier de charge

En gros le cahier de charge est composé des éléments suivants :

- Contexte et périmètre du projet.
- Spécifications non fonctionnelles.
- Spécifications fonctionnelles.
- Ressources.
- Délais.
- Besoins financiers et budget.

## II. Etude de faisabilité et analyse des besoins

### Cahier de charge

<https://openclassrooms.com/fr/courses/6739646-realisez-un-cahier-des-charges-fonctionnel>

**==: Certification obligatoire (cahier de charges fonctionnelles)**

# III. Architectural Patterns

## Patterns

Un pattern décrit à la fois un problème qui se produit très fréquemment dans l'environnement et l'architecture de la solution à ce problème de telle façon que l'on puisse utiliser cette solution des milliers de fois sans jamais l'adapter deux fois de la même manière.

Les patterns sont des composants logiques décrits indépendamment d'un langage donné (solution exprimée par des modèles semi-formels).

# III. Design Patterns

## Patterns

Pour mener à bien le développement on utilise généralement des Patterns. Il existe plusieurs catégories de Patterns :

- **Architectural Patterns**
  - Schémas d'organisation structurelle de logiciels (pipes, filters, brokers, blackboard, MVC, ....)
- **Design Patterns**
  - Patron de conception, caractéristiques clés d'une structure de conception commune à plusieurs applications.
- **Idioms ou coding Patterns**
  - Solution liée à un langage
- **Anti-patterns**
  - démarche pour sortir d'une mauvaise solution
- **Organizational Patterns**
  - Schémas d'organisation de tout ce qui entoure le développement d'un logiciel (humains)



# III. Design Patterns

## **MVC (Model View Controller) ou MVT (Model View Template)**

La complexité des projets logiciels oblige de suivre une démarche pour atteindre un certain nombre d'objectifs.

Il s'agit ici de présenter le design de conception MVC autour des points:

- Principes théoriques
- Mise en œuvre du MVC & MVT
  - Framework Php Laravel, Symfony
  - Framework Python Django, Flask

# III. Design Patterns

## Principes théoriques du MVC

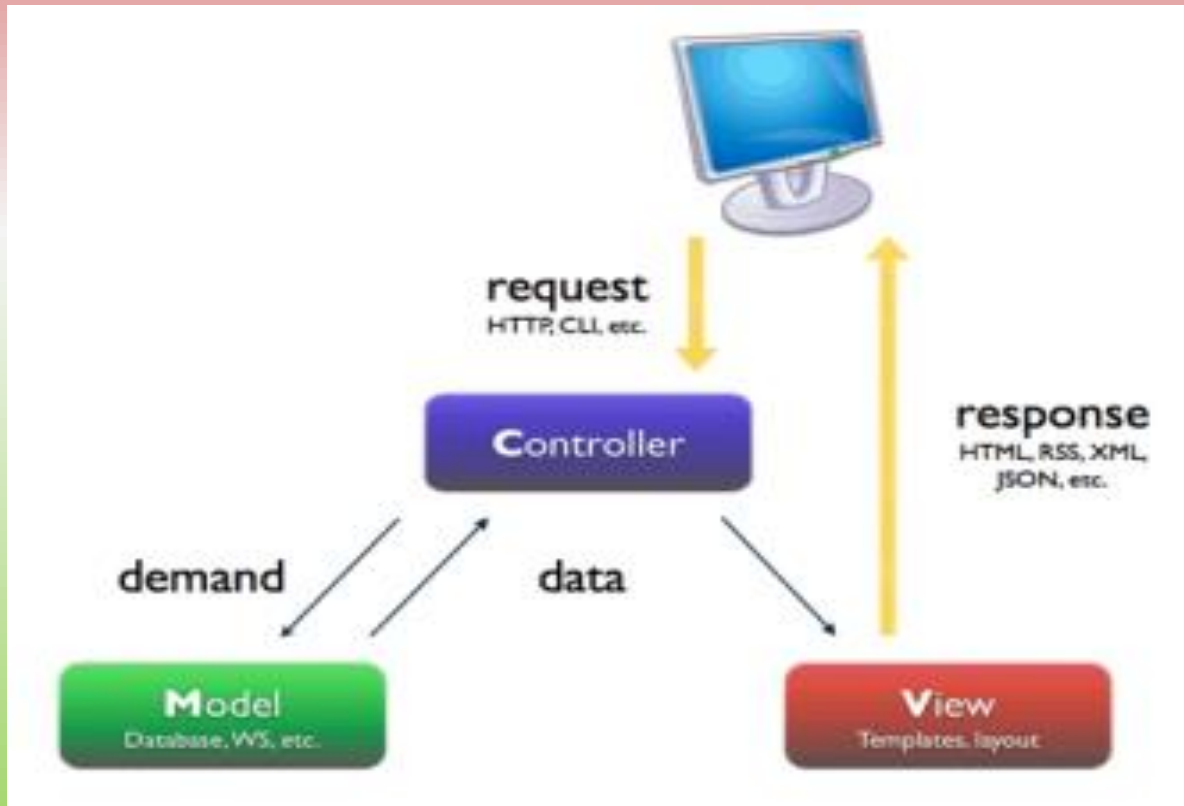
Le modèle MVC indique une manière d'architecturer une application en la décomposant en trois parties : le **M**odèle, la **V**ue et le **C**ontrôleur.

## Principes théoriques du MVT

Le modèle MVT indique une manière particulière d'architecturer une application en la décomposant en trois parties : le **M**odèle, la **V**ue et le **T**emplate.

# III. Design Patterns

## Principes théoriques du MVC



le **Modèle** : fournit des services au contrôleur pour lui permettre de préparer les données à afficher. **Étape 2**

la **Vue** : est chargée d'afficher les données fournies par le contrôleur à l'utilisateur. **Étape 3**

le **Contrôleur** : réceptionne et interprète la demande de l'utilisateur (une requête HTTP par exemple). **Étape 1**

# III. Design Patterns

## Principes théoriques du MVC

Le **M**odèle contient la partie logique métier ainsi que l'accès aux données. Généralement c'est un ensemble de fonctions pour un modèle procédural ou de classes pour un modèle orienté objet. Ainsi le but étant de gérer l'organisation des données. Chacune de ces fonctions ou classe effectuera une action bien précise. Les requêtes SQL à des SGBD se font dans ces fonctions ou classes.

# III. Design Patterns

## Principes théoriques du MVC

La **Vue** est la partie qui s'occupe des interactions avec l'utilisateur : présentation, saisie et validation des données. Elle est la seule qui a le droit de contenir du HTML.

Le **Contrôleur** gère la dynamique de l'application. Elle fait le lien entre l'utilisateur et le reste de l'application.

# III. Design Patterns

**MVC (Model View Controller) ou MVT (Model View Template)**

<https://openclassrooms.com/fr/courses/4425066-concevez-un-site-avec-flask>

# IV. Conception de la solution

Dans cette partie, il s'agit de la modélisation de notre futur logiciel qui représente une étape très importante.

Objectif de cette tâche : description de la manière et les outils

- Conception architecturale, modélisation de haut niveau : modules
- Conception détaillé, modélisation bas niveau : on se rapproche du code
- Caractéristiques/ Contraintes particulières : persistance, composants sur l'étagère, langages/paradigmes

# IV. Conception de la solution

## **Cycle de vie du logiciel**

Un cycle de vie propose une stratégie qui décrit la manière dont les activités sont organisées.

Les modèles du cycle de vie du logiciel sont des plans de travail qui permettent de planifier le développement.



# IV. Conception de la solution

## Plusieurs modèles de cycle de vie :

- Le modèle du code-and-fix;
- Le modèle de la transformation automatique;
- Le modèle de la cascade;
- Le modèle en V;
- Le modèle par incrément;
- Le modèle par prototypage;
- Le modèle en spirale.

# IV. Conception de la solution

## **Plusieurs langages existent pour modéliser.**

Un seul a réussi à s'imposer comme le langage pour la programmation orienté objet : il s'agit UML (Unified Modeling Language)

- Il est standardisé par l'Open Management Group,
- mondialement utilisé (plus de 80% des projets),
- bien outillé et documenté (livres, tutoriels, mooc, ...)

# IV. Conception de la solution

- UML est le langage de modélisation orienté objet le plus connu et le plus utilisé,
- UML n'est pas une méthode mais plutôt un langage de modélisation utilisé par la méthode UP(Unified Processus)
- En UML, les informations sont représentées sous forme de diagrammes

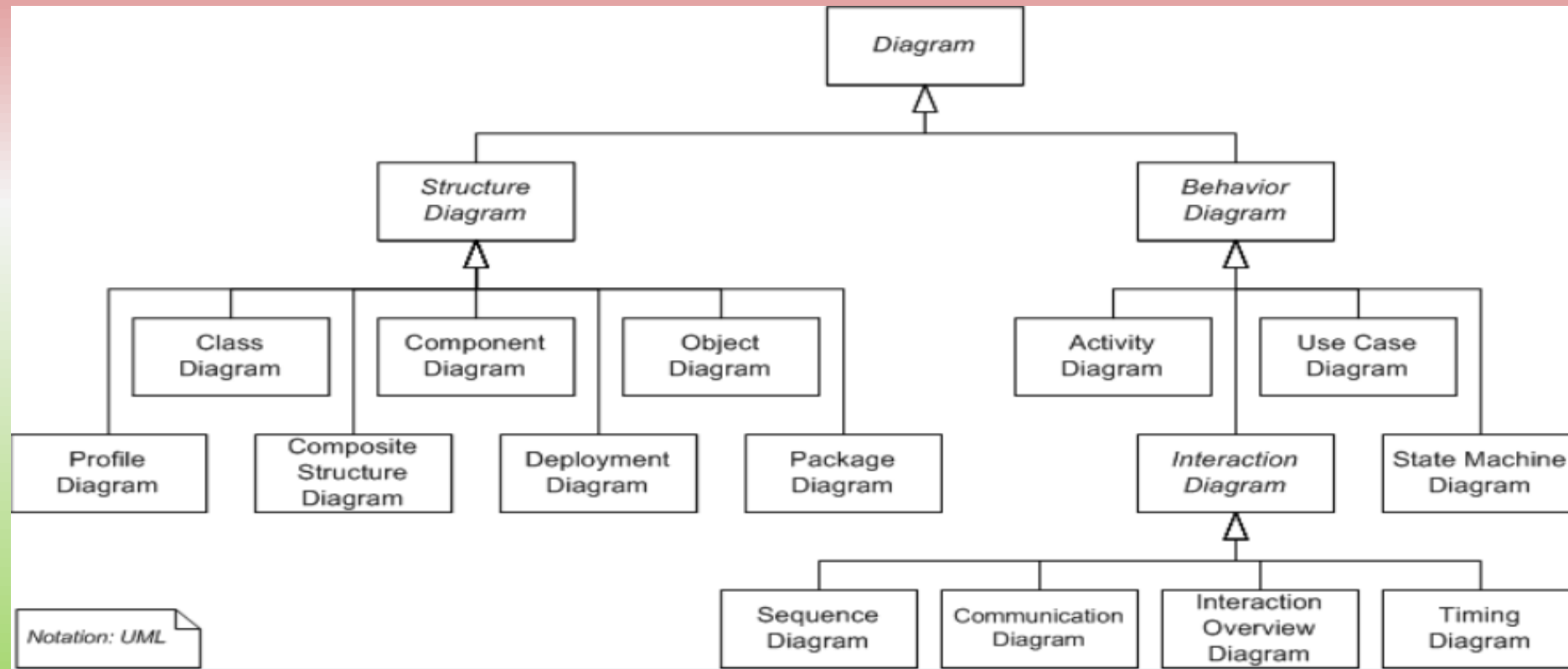
# IV. Conception de la solution

UML couvre les étapes suivantes du cycle de vie du logiciel

- **Expressions des besoins,**
- **Analyse,**
- **Conception,**
- **Réalisation,** lorsqu'il est considéré comme langage de programmation
- Validation,
- **Déploiement ,**
- Maintenance.

# IV. Conception de la solution

## UML 2.5



# IV. Conception de la solution

## **Merise (Données, Traitement)**

Modèle Conceptuel de Données

Modèle Logique de Données

Modèle Physique de Données

Modèle Conceptuel de Traitement

Modèle Logique de Traitement

TP avec l'atelier de génie logiciel Windesign

# IV. Conception de la solution

## Choix de l'outil technique

- Choix du paradigme (procédural, objet, fonctionnel, composant, aspect, rôle, ...)
- Choix du langage (C, Java, C++, Python, ...)
  - Simplicité
  - Performances temps de calcul et mémoire
  - Mécanismes
  - Fonctionnalités des librairies/packages standards associés

Ces choix dépendent du contrat, des besoins, de la conception, des compétences humaines.

# V. Les ateliers de génie logiciel

Ce sont des outils aidant à la réalisation de logiciels. Ainsi ils servent à couvrir le cycle de vie du logiciel (Analyse; conception; réalisation; maintenance; ...).

Les AGL intègrent différents outils (les « outils CASE ») d'aide au développement de logiciels. Ainsi on distingue :

- Les outils verticaux intervenant durant tout le processus logiciel;
- Les outils horizontaux intervenant à des phases du cycle de vie du logiciel;
- Reversing (par exemple nous avons rose qui génère un code pendant que d'autres font l'inverse en générant les diagrammes à partir d'un code).

Exemples :PowerAMC, Win Design, WinDev, Eclipse, NetBeams, Unix/Linux ...



# V. Les ateliers de génie logiciel

<https://boowiki.info/art/conception-de-logiciels/design-genie-logiciel.html>

Ateliers de production de jeux vidéos

<https://www.monemploi.com/formations/programmes/universitaire/informatique/conception-de-jeux-video>

<https://www.monemploi.com/metiers-et-professions/fiche/testeur-de-jeux-videos>

<https://openclassrooms.com/fr/courses/5249006-concevez-une-interface-cliquable/5778836-realisez-un-prototype>

# VI. Méthodes agiles

## 1. Les limites de l'approche traditionnelle

La méthode classique est une succession de phases qui se déverse les unes dans les autres. Le problème est qu'on ne peut pas commencer une phase de l'approche classique tant que la précédente n'est pas terminée. Alors le projet suit un plan déjà connu : la conception, la réalisation et la recette.

Souvent, lors de la phase de réalisation, on parle d'**effet tunnel** puisque le commanditaire / MOA du projet en général ne voit pas grande chose de concret sur le projet en cours tant que l'on n'arrive pas à la phase de recette. C'est ainsi que cette approche a rapidement montré ses limites avec la complexité grandissante des projets liés à trois principaux facteurs qui sont le travail en équipe, la technologie, l'anticipation des besoins utilisateur.

# VI. Méthodes agiles

## 2. Les différentes méthodes Agiles

Les méthodes agiles sont des groupes de pratiques de pilotage et réalisation de projets. Elles ont pour origine le **manifeste Agile**, rédigé en 2001, qui consacre le terme d' « agile » pour référencer de multiples méthodes existantes.

# VI. Méthodes agiles

## 2. Les différentes méthodes Agiles

Les quatre (4) valeurs du manifeste Agile :

Nous découvrons de meilleures approches pour faire du développement logiciel, en faisant nous-même et en aidant les autres à faire.

Grâce à ce travail, nous en sommes arrivés à préférer et favoriser :

Les individus et leurs interactions	plus que les processus et les outils.
Un logiciel qui fonctionne	plus qu'une documentation exhaustive.
La collaboration avec les clients	plus que la négociation contractuelle.
L'adaptation au changement	plus que le suivi d'un plan.

Cela signifie que bien qu'il y ait de la valeur dans les éléments situés à droite, notre préférence se porte sur les éléments qui se trouvent sur la gauche.

# VI. Méthodes agiles

## 2. Les différentes méthodes Agiles

Les douze (12) principes du manifeste Agile :

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
5. Réalisez les projets avec de personnes motivées. Fournissez-leur l'environnement et le soutien dont elles ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
6. Privilégiez la co-location de toutes les personnes travaillant ensemble et le dialogue en face à face comme méthode de communication.

# VI. Méthodes agiles

## 2. Les différentes méthodes Agiles

Les douze (12) principes du manifeste Agile :

7. Un logiciel opérationnel est la principale mesure de progression d'un projet.
8. Les processus agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
9. Une attention continue à l'excellence technique et à un bon design.
10. La simplicité. Autrement dit, l'art de minimiser la quantité de travail inutile est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
12. À intervalles réguliers, l'équipe réfléchit aux moyens possibles pour devenir plus efficace. Puis elle s'adapte et modifie son mode de fonctionnement en conséquence.

# VI. Méthodes agiles

## 2. Les différentes méthodes Agiles

Une fois qu'une organisation décide d'adopter une gestion de développement Agile, il reste encore à choisir la méthodologie la plus adaptée à son projet. En effet, les méthodes Agiles disponibles sont nombreuses et peuvent être source de confusion :

Scrum	Agile Unified Process (Agile UP ou AUP)
Kanban	Crystal
Feature Driven Development	Dynamic Systems Development Method (DSDM)

# VI. Méthodes agiles

## 3. Intégration continue ou Continuous Integration (CI), une méthodologie du génie logiciel

Son objectif est d'éviter les conséquences d'un « effet tunnel » entre les phases de développement et d'intégration d'un cycle en V traditionnel.

Dans la pratique, il s'agit de fusionner « régulièrement » le code produit par les membres de l'équipe de développement afin de s'assurer tout aussi régulièrement de la qualité du logiciel résultant.



# VI. Méthodes agiles

## 3. Intégration continue ou Continuous Integration (CI), une méthodologie du génie logiciel

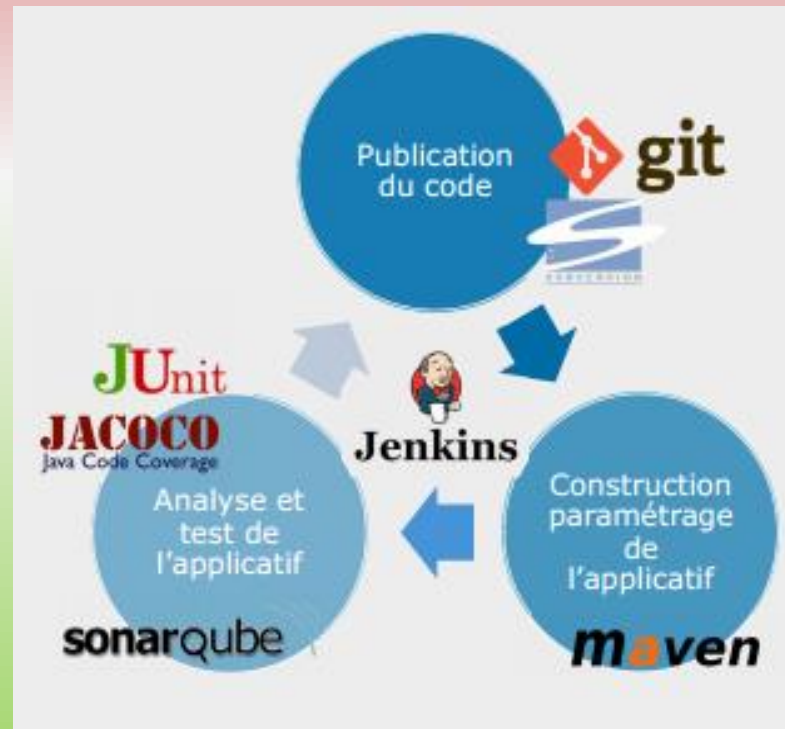
Les pratiques associées à l'IC se sont étendues depuis l'apparition du concept.

En plus de la notion d'intégration en parallèle de développements, lorsqu'on parle d'IC, cela concerne désormais aussi les aspects d'automatisation :

- De la vérification de la qualité du code (et de sa documentation)
- Des tests
- De la vérification de la sécurité du code
- Du packaging
- Du déploiement (extension vers la démarche DevOps)

# VI. Méthodes agiles

## 3. Intégration continue ou Continuous Integration (CI), une méthodologie du génie logiciel



# VII. Implémentation logicielle

Edition du code source

Déploiement de la solution (Outils du Devops)

<https://openclassrooms.com/fr/courses/7162856-gerez-du-code-avec-git-et-github> github

Laravel, Windev, Django, deploiement moodle, application mobile avec flutter

<https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql>

<https://openclassrooms.com/fr/courses/7008001-debutez-avec-react>

<https://openclassrooms.com/fr/courses/5489551-creez-un-site-moderne-et-professionnel-avec-wordpress-5>

<https://openclassrooms.com/fr/courses/7471261-debutez-avec-angular>

# VII. Implémentation logicielle

## Initiation à docker

Docker crée des outils simples et une approche de packaging universelle qui regroupe toutes les dépendances d'application dans un conteneur qui est ensuite exécuté sur Docker Engine.

Docker Engine permet aux applications conteneurisées de s'exécuter partout et de manière cohérente peut importe l'infrastructure, résolvant « l'enfer des dépendances » pour les développeurs et les équipes d'exploitation.

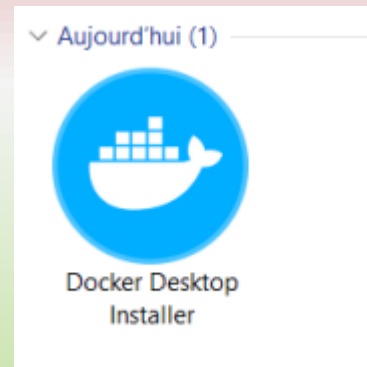
# VII. Implémentation logicielle

## Travaux Pratiques docker

### Télécharger et installer Docker Desktop

Se rendre sur <https://www.docker.com/>

Une fois téléchargé, on obtient ce qui suit :



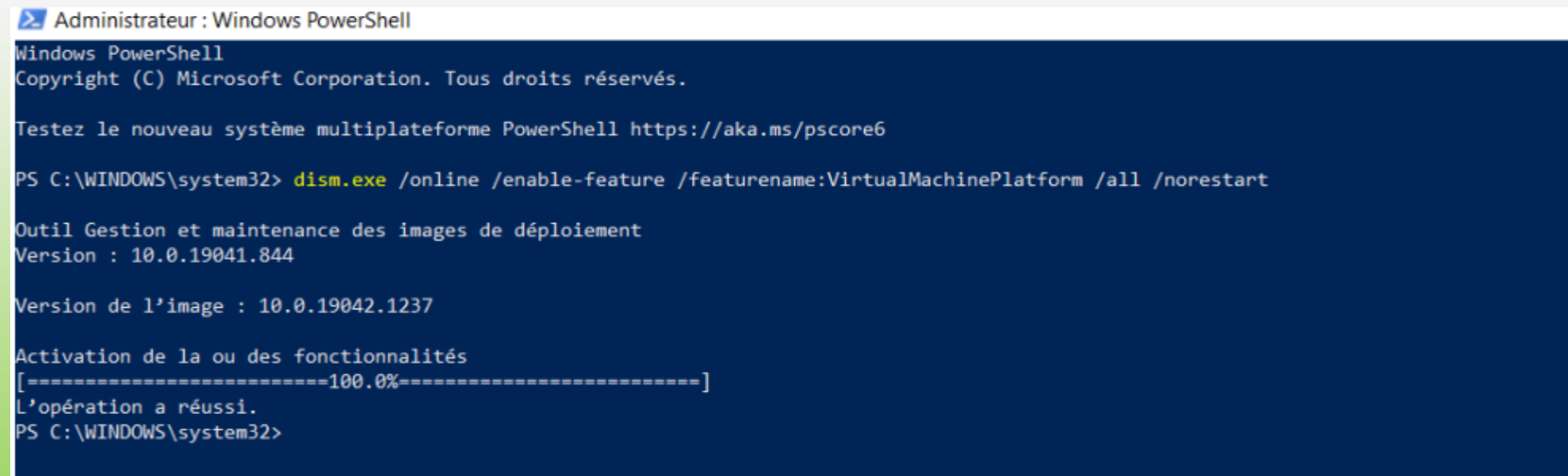
# VII. Implémentation logicielle

## Travaux Pratiques docker

### Télécharger et installer Docker Desktop

Lors de l'installation cocher la case pour l'installation de windows components for WSL 2

Après l'installation si la composante n'est toujours pas installée; executer le code suivant comme suite :



```
Administrateur : Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

Outil Gestion et maintenance des images de déploiement
Version : 10.0.19041.844

Version de l'image : 10.0.19042.1237

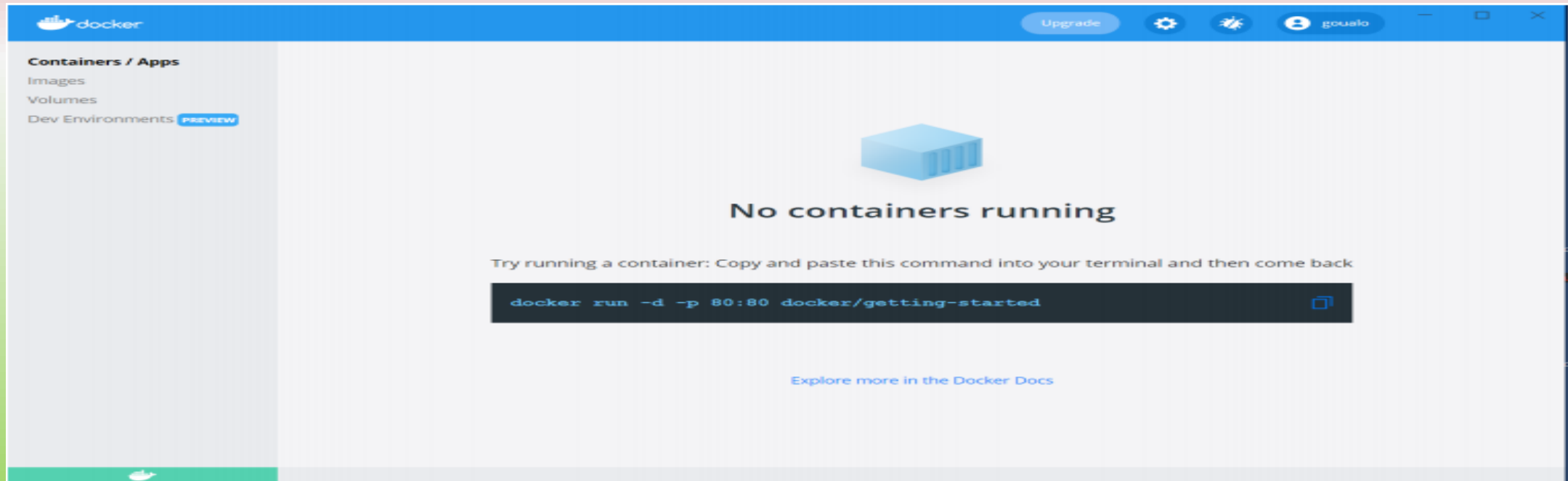
Activation de la ou des fonctionnalités
[=====100.0%=====]
L'opération a réussi.
PS C:\WINDOWS\system32>
```

# VII. Implémentation logicielle

## Travaux Pratiques docker

### Télécharger et installer Docker Desktop

Si tout se passe bien, vous obtenez ce qui suit :



# VII. Implémentation logicielle

## Travaux Pratiques docker

### Exécuter votre premier programme

`docker run -d -p 80:80 docker/getting-started`

ou `docker run -dp 80:80 docker/getting-started`

```
PS C:\WINDOWS\system32> docker run -dp 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
540db60ca938: Pull complete
0ae30075c5da: Pull complete
9da81141e74e: Pull complete
b2e41dd2ded0: Pull complete
7f40e809fb2d: Pull complete
758848c48411: Pull complete
23ded5c3e3fe: Pull complete
38a847d4d941: Pull complete
Digest: sha256:10555bb0c50e13fc4dd965ddb5f00e948ffa53c13ff15dc85b7ab65e1f240b
Status: Downloaded newer image for docker/getting-started:latest
1faf8a3b3bb5ab00f971a54321d2576ec2233ad646c7037d216a2dfb6123e61f
PS C:\WINDOWS\system32>
```

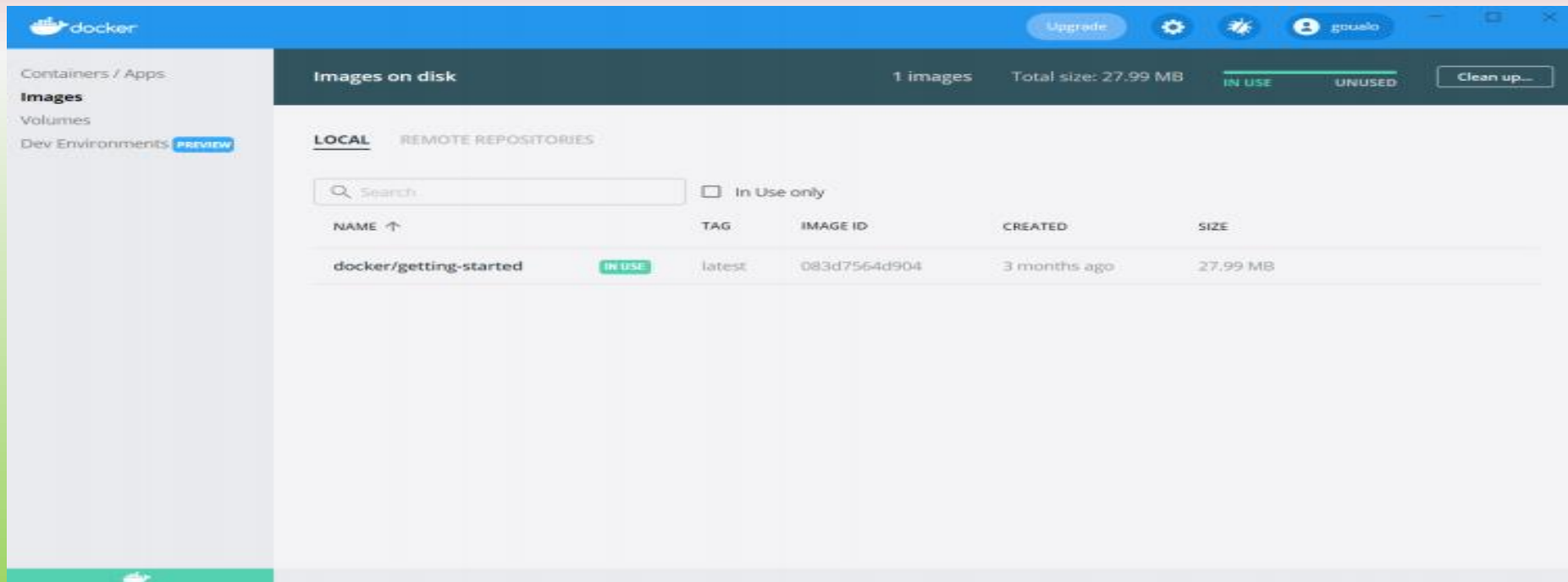


# VII. Implémentation logicielle

## Travaux Pratiques docker

### Exécuter votre premier programme

Une fois l'exécution terminée, vous obtenez l'image dans votre docker :



# VII. Implémentation logicielle

## Travaux Pratiques docker

Télécharger et exécuter une autre image (un jeu pour vous amusez)

```
docker run -dp 8080:80 alexwhen/docker-2048
```

```
PS C:\Users\vicorlen.bobet> docker run -dp 8080:80 alexwhen/docker-2048
Unable to find image 'alexwhen/docker-2048:latest' locally
latest: Pulling from alexwhen/docker-2048
Image docker.io/alexwhen/docker-2048:latest uses outdated schema manifest format. Please upgrade to a schema2 image for better future compatibility. More information at https://docs.docker.com/registry/spec/deprecated-schema-v1/
c862d82a67a2: Pull complete
a3ed95caeb02: Pull complete
69dbbd8c451d: Pull complete
e9b345a0f742: Pull complete
Digest: sha256:4913452e5bd092db9c8b005523127b0f62821867021e23a9acb1ae0f7d2432e1
Status: Downloaded newer image for alexwhen/docker-2048:latest
f1c8eda082dffa0cb6c6d6147145eba62aa1fc853b5bffc40eab455dc5bda531
PS C:\Users\vicorlen.bobet> docker ps
```

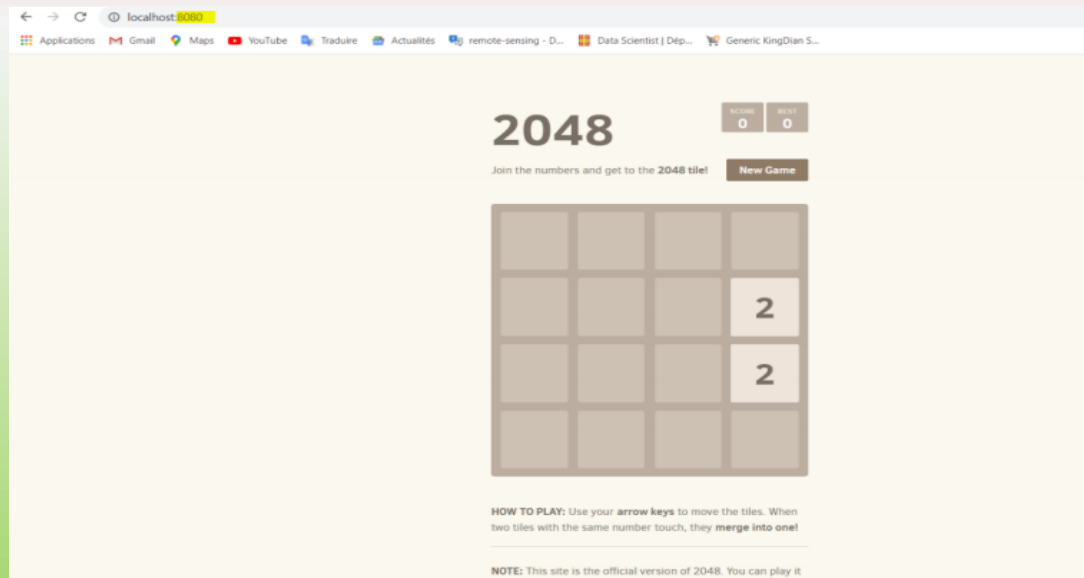
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f1c8eda082df	alexwhen/docker-2048	"nginx -g 'daemon of..."	About a minute ago	Up About a minute	0.0.0.0:8080->80/tcp, :::8080->80/tcp	great_moore
1faf8a3b3bb5	docker/getting-started	"/docker-entrypoint..."	52 minutes ago	Up 52 minutes	0.0.0.0:80->80/tcp, :::80->80/tcp	hungry_helsenberg

# VII. Implémentation logicielle

## Travaux Pratiques docker

Télécharger et exécuter une autre image (un jeu pour vous amusez)

Ouvrez votre navigateur et saisissez localhost:8080 et bim !



# VII. Implémentation logicielle

## Gestion de versions (ou version control) avec Git

La gestion de versions consiste à gérer l'ensemble des versions d'un ou plusieurs fichiers.

Les logiciels évoluant, chaque étape d'avancement est appelée version (ou révision).

Une modification du code est une évolution entre deux versions.

Il existe de nombreux logiciels capables d'aider le développeur dans cette tâche (par exemple Git)

# VII. Implémentation logicielle

## Gestion de versions (ou version control) avec Git

Ainsi ces logiciels permettent :

- La sauvegarde (avec ou sans serveur distant)
- La conservation de l'historique des fichiers (l'auteur des modifications, la date, une note explicative de cette modification)
- Voir les changements
- Revenir aux versions précédentes
- Collaborer.

# VII. Implémentation logicielle

## Gestion de versions (ou version control) avec Git

### Répertoire de travail

On appelle répertoire de travail (working copy en anglais) les fichiers effectivement présents dans le répertoire géré par Git.

### Un dépôt

Un dépôt (repository) est un dossier contenant l'ensemble des données associées au projet (les versions et les modifications d'un projet).

**git init** pour la création d'un dépôt local

# VII. Implémentation logicielle

## Gestion de versions (ou version control) avec Git

### Commit

Sauvegarder au sens de git, c'est effectuer un commit.

### Etat

La commande **git status** permet de connaître l'état d'un fichier du répertoire de travail vis-à-vis du dernier commit.

### Historique git

La commande **git log** permet de voir l'ensemble des commit.

# VII. Implémentation logicielle

**Gestion de versions (ou version control) avec Git**

<https://openclassrooms.com/fr/courses/7688581-devenez-un-expert-de-git-et-github>

**==: Certification obligatoire (git et github)**



# VII. Implémentation logicielle

## CI/CD : l'intégration et la livraison en continue

CI/CD représente pour le Devops un outil de modernisation des applications.

<https://openclassrooms.com/fr/courses/2035736-mettez-en-place-lintegration-et-la-livraison-continues-avec-la-demarche-devops>

**==: Certification obligatoire (pipeline d'intégration et de livraison continue)**

# VIII. Qualité logicielle

La qualité d'un logiciel s'évalue à deux niveaux :

- **Lors de l'utilisation**

- fiabilité (correction et robustesse),
- adéquation aux besoins (y compris aux besoins implicites !),
- ergonomie (simplicité et rapidité d'emploi, personnalisation),
- efficacité,
- convivialité,
- faible coût et respect des délais bien entendu,
- etc.

# VIII. Qualité logicielle

La qualité d'un logiciel s'évalue à deux niveaux :

- **Lors de la maintenance**

- Flexible
- Portable
- Structuré
- Une indépendance maximum entre les structures
- Documenté

N.B: Ces différentes qualités ne sont pas toujours compatibles ni même réalisables. Néanmoins il faut trouver des compromis.

# VIII. Qualité logicielle

Les conséquences lorsque la qualité fait défaut:

- Indisponibilité du logiciel, du matériel
- Difficulté de maintenir l'application
- Retard de livraison
- Difficulté d'utilisation
- Abandon de l'application

# VIII. Qualité logicielle

## Tests logiciels

Les tests permettent de valider le fonctionnement, ou révéler des dysfonctionnements.

Un test logiciel permet aussi de vérifier s'il répond aux exigences du client.

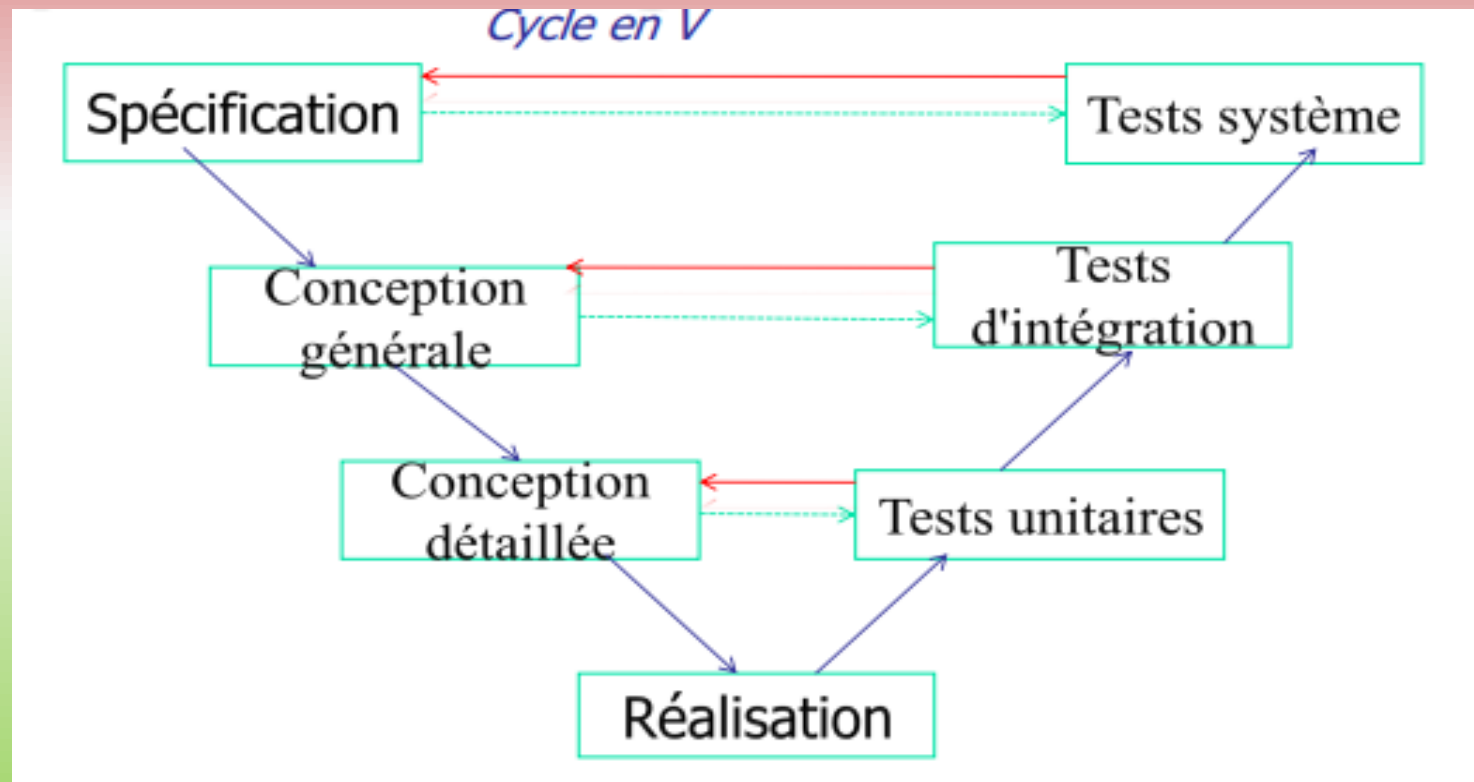
# VIII. Qualité logicielle

## Principes de tests

1. Un programmeur ne doit pas tester ses propres programmes.
2. Ne pas faire les tests en pensant qu'aucune erreur ne sera trouvée(un programme qui test doit retourné par défaut une erreur).
3. Il faut chercher à connaitre les résultats attendus avant l'exécution du programme.
4. Il faut vérifier attentivement les résultats de chaque test ainsi que leur pertinence.
5. Il faut également bien tester avec des programmes de tests valides ou invalides.
6. Vérifier ce que fait le programme lorsqu'il n'est pas censé le faire.

# VIII. Qualité logicielle

## Les différents tests pour un cycle de vie logiciel en V



# VIII. Qualité logicielle

## Test unitaire

Un test unitaire permet de vérifier qu'une unité (ou partie) du code ne contienne pas d'erreur. Il faut donc bien "isoler" la partie (classe) testée.

Il existe des frameworks dans chaque langage :

- Junit pour Java
- Cunit, cppUnit, ... pour C++
- Nunit, ... pour C#
- unittest, ... pour Python



# VIII. Qualité logicielle

## Test unitaire

### Avantages

- Gain de temps sur l'ensemble du projet final
- La reprise du code par une autre personne est plus facile
- Le code final est plus efficace
- Garantie la non régression
- Détection de bug plus facilement
- Aide à isoler les fonctions

### Inconvénients

- Impression de perdre du temps car beaucoup de tests
- Demande beaucoup de pratique pour des tests efficaces

# VIII. Qualité logicielle

## Test unitaire

<https://openclassrooms.com/fr/courses/7155841-testez-votre-projet-python>

# IX. Gestion de la configuration

Ce domaine s'intéresse à l'identification de tous les composants d'un logiciel, incluant les différents documents et plans afin d'être en mesure dans le temps de gérer d'une façon systématique les changements apportés à ceux-ci, tout en conservant leur intégrité.

Un gestionnaire de configuration logiciel (GCL) est un dépôt de données qui permet de gérer la chronologie des différentes versions d'un produit.

Un GCL facilite le travail collaboratif. Plusieurs personnes peuvent en effet travailler en parallèle, et mettre à jour leurs répertoires de travail locaux en fonction des modifications apportées par les autres.

# X. Maintenance logicielle

La maintenance logicielle s'intéresse aux activités de modifications du logiciel suite à sa livraison pour effectuer des correctifs et des améliorations.

La maintenance assure les évolutions, l'adaptation.

Deux techniques de maintenance

- Correction des erreurs du système
- Demande d'évolution (modification de l'environnement technique, nouvelles fonctionnalités)

# XI. Gestion de projets logiciels

La gestion de projets selon le cycle de vie du projet :

## **Au démarrage du projet (phase de lancement, d'initiation)**

- Plan d'Assurance Qualité (PAQ) + Planification initiale + Plan de gestion des risques initial
- Initialisation des tableaux de bord

## **Pendant le projet, d'un point de vue production**

- Le suivi de l'avancement (activités de l'équipe et planning)
- Comités de pilotage et suivis de projet
- Mise à jour du plan de gestion des risques

# XI. Gestion de projets logiciels

## **Pendant le projet, d'un point de vue production**

- Gestion des anomalies et des évolutions
- Suivi financier
- Suivi de la satisfaction client
- Gestion des livraisons et de la configuration
- Suivi des tests
- Suivi des procès verbaux de recette

## **Pendant le projet, d'un point de vue financier**

- Suivi du budget formations
- Suivi du budget achats

# XI. Gestion de projets logiciels

## **Pendant le projet, d'un point de vue financier**

- Suivi du budget sous-traitance
- Suivi du chiffre d'affaire et de la marge générés
- Suivi de la facturation et des écritures comptables
- Suivi de l'échéancier de paiement et des règlements
- Suivi des besoins en formation
- Suivi de carrière (promotion ou augmentation salaire)

## **En fin projet**

- Le bilan de projet

# Bibliographie & Webographie

<https://www.cairn.info/le-genie-logiciel--9782130548928-page-16.htm>

<https://openclassrooms.com/fr/courses/6739646-realisez-un-cahier-des-charges-fonctionnel>  
cahier de charge fonctionnel

<https://openclassrooms.com/fr/courses/4810836-decouvrez-le-cloud-avec-amazon-web-services>  
cloud aws

<https://openclassrooms.com/fr/courses/5647281-appliquez-le-principe-du-domain-driven-design-a-votre-application?archived-source=2035826>

## UX Design

<https://openclassrooms.com/fr/courses/3013856-ux-design-decouvrez-les-fondamentaux>

<https://designinteractif.gobelins.fr/2018/10/09/comment-integrer-le-design-ux-dans-les-methodes-de-genie-logiciel/>

<https://lagrandeourse.design/blog/outils/les-outils-utiles-pour-travailler-sur-un-projet-ux/>

<https://www.olloweb.com/fr/creation-graphique-web/conception-interface-graphique.html> cms



# Bibliographie & Webographie

<https://openclassrooms.com/fr/courses/3936801-composez-des-interfaces-utilisateurs-en-material-design/3936808-comprenez-la-notion-dinterface-utilisateur>

<https://openclassrooms.com/fr/courses/4568596-construisez-une-interface-utilisateur-flexible-et-adaptative>

<https://openclassrooms.com/fr/courses/5249021-initiez-vous-a-la-methode-atomic-design>

<https://openclassrooms.com/fr/courses/5454886-creez-et-appliquez-une-charte-graphique/5715676-prenez-en-compte-les-utilisateurs-finaux>

<https://openclassrooms.com/fr/courses/6227506-assistez-la-maitrise-douvrage-dun-projet-si/6796640-concevez-votre-ihm> maitrise d'ouvrage