

Projet de Génie Logiciel S-INFO-015 et S-INFO-106

Tom Mens

Service de Génie Logiciel – Département Informatique
informatique.umons.ac.be/genlog



Projet de Génie Logiciel

Travail en *groupes de 4 (ou 3) personnes*

En 2 phases:

- **S-INFO-015** (Q1) Projet d'analyse et de conception
 - Pour BA2 Sciences Informatiques
- **S-INFO-106** (Q2) Projet de développement logiciel
 - BA2 Sciences Informatiques
 - Master Bloc complémentaire Sciences Informatiques
 - Bachelier Mathématiques (en option)
 - **Avec défense orale**

Projet de Génie Logiciel

- Rapports à rendre sur moodle.umons.ac.be
 - selon les échéances prévues
- Attention aux **critères de recevabilité!**
 - Un projet non-recevable ne sera pas évalué
- Évaluation sur base de la **qualité** des *livrables* rendus

Projet de génie logiciel = projet de groupe



students



Projet de génie logiciel = projet de groupe



groupe

fonctionnalité
de base



students



Le poids (pondération) de la fonctionnalité de base et des extensions dépendra de la taille du groupe:

- Pour un groupe de 4: 60% pour la partie commune; 40% pour l'extension individuelle
- Pour un groupe de 3: 70% pour la partie commune; 30% pour l'extension individuelle

Pour la phase d'**analyse de conception ET**
pour la phase de **développement logiciel**

extension D



Q1 - Projet d'analyse et de conception

S-INFO-015

Lundi 26/9/2022

Présentation du projet et distribution de l'énoncé

Lundi 10/10/2022

Date limite pour la formation des groupes et choix d'extensions

Début officiel de la phase d'analyse et de conception

Vendredi 16/12/2022

Remise des livrables:

- Rapport de modélisation: modèles UML + modèle de données (ERD) + design de l'API REST
- Maquette de l'interface utilisateur

Q2 - Projet de développement logiciel

S-INFO-106

Lundi 6/2/2023

Début officiel de la phase de développement logiciel

Lundi 20/3/2023

Remise d'un prototype fonctionnelle présentant un proof-of-concept des briques technologiques choisies, illustré avec (une partie de) les ***fonctionnalités de base***

Lundi 24/4/2023

Remise finale des livrables:

- Rapport d'implémentation
- Vidéo de la fonctionnalité (manuel d'utilisation)
- Scripts gradle, code source, tests unitaires, archives auto-exécutables, JavaDoc

Mercredi 18/5/2023

Défense orale des projets

Présence obligatoire!

Énoncé du projet

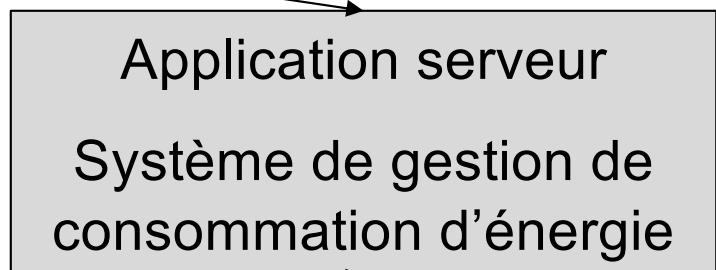
Gestion de portefeuille financier



- Le projet consiste à réaliser en groupe un système logiciel de gestion de portefeuille d'énergie et d'approvisionnement énergétique pour des habitations. Bien que le projet fasse partie d'un cours universitaire, son contenu correspond à des besoins réels, il ne s'agit donc pas juste d'un sujet artificiel. La qualité du projet (à la fois au niveau du code source, des tests unitaires, de la fonctionnalité et de la convivialité) est donc essentielle.
- Le projet doit suivre un modèle **client-serveur** avec une **architecture à trois couches**, afin d'assurer une *séparation de préoccupations* entre la **couche de presentation** (les applications clients), et la **couche logique** ainsi que la **couche de données** hébergées sur le **serveur**.
- Le **frontend** (clients) sera développé comme deux applications web utilisant la technologie **JavaScript**: une application *consommateur* pour la gestion de portefeuilles d'énergie; une application *fournisseur* pour la gestion de contrats, compteurs, clients, etc.
- Le **backend** (serveur) sera développé en **Java** et utilisera une base de données. L'interaction entre frontend et backend se fera par une API REST.

Énoncé du projet

Gestion de portefeuille financier



Énoncé du projet

Architecture client-serveur



Application web “cliente” pour consommateurs

API REST

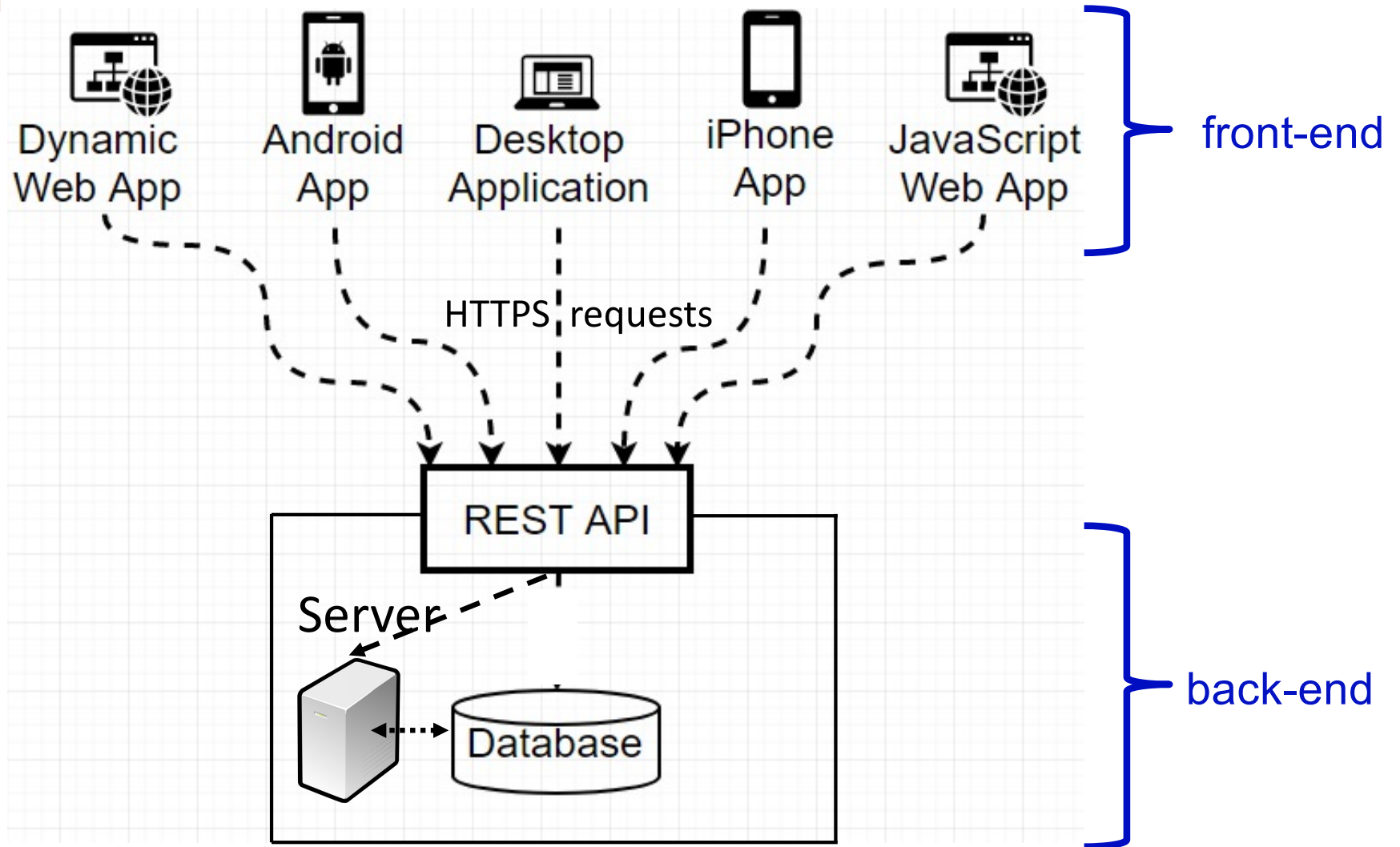
Application “serveur”
+ base de données

API REST

Application web “cliente” pour fournisseurs



Front-end versus back-end



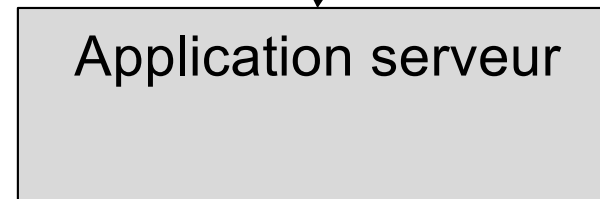
Architecture à 3 couches (séparation de préoccupations)

Couche de
présentation



front-end

Couche
logique



back-end

Couche de
données

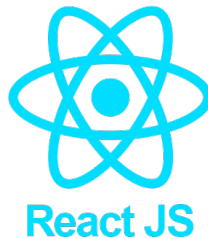


SQL / ORM / ...

REST



Architecture à 3 couches (séparation de préoccupations)



front-end



back-end



Test-Driven Development (TDD)

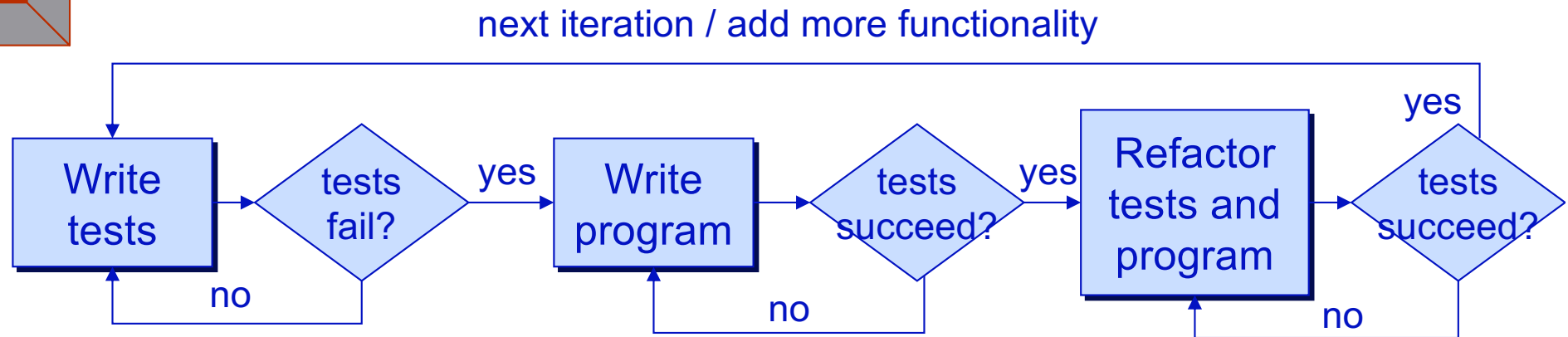
- L'utilisation des **tests unitaires** est obligatoire
- Un processus de **test-driven development** doit être suivi

« Test-driven development: Concepts, taxonomy and future directions ». David Janzen, Hossein Saiedian, IEEE Computer, September 2005, pp. 43-50

*“Test-driven development (TDD) is the craft of producing automated tests for production code, and using that process to **drive design** and **programming**. For every tiny bit of functionality in the production code, you first develop a test that specifies and validates what the code will do. You then produce exactly as much code as will enable that test to pass. Then you refactor (simplify and clarify) both the production code and the test code.”*

The Agile Alliance

Test-Driven Development (TDD)

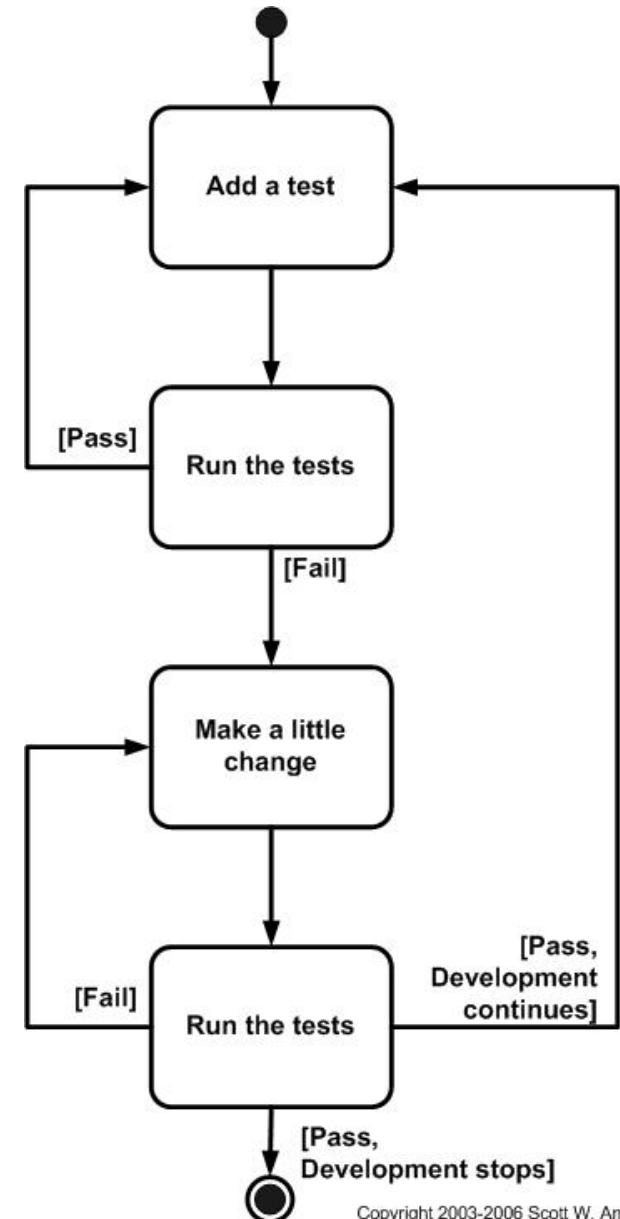
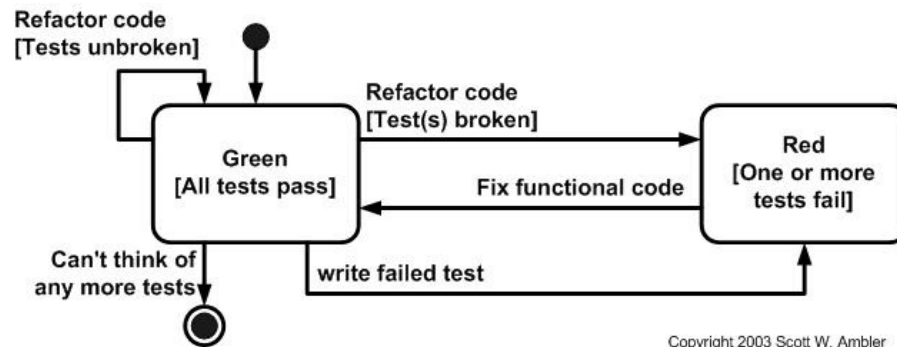


“TDD is **not about testing**, it is about **design**”

- écrire les tests avant d’écrire le code source oblige le développeur de penser à comment utiliser le code avant de réaliser l’implémentation
- Automated tests = active documentation
 - automatiser l’exécution des tests permet de répéter tous les tests après chaque modification

Test-Driven Development (TDD)

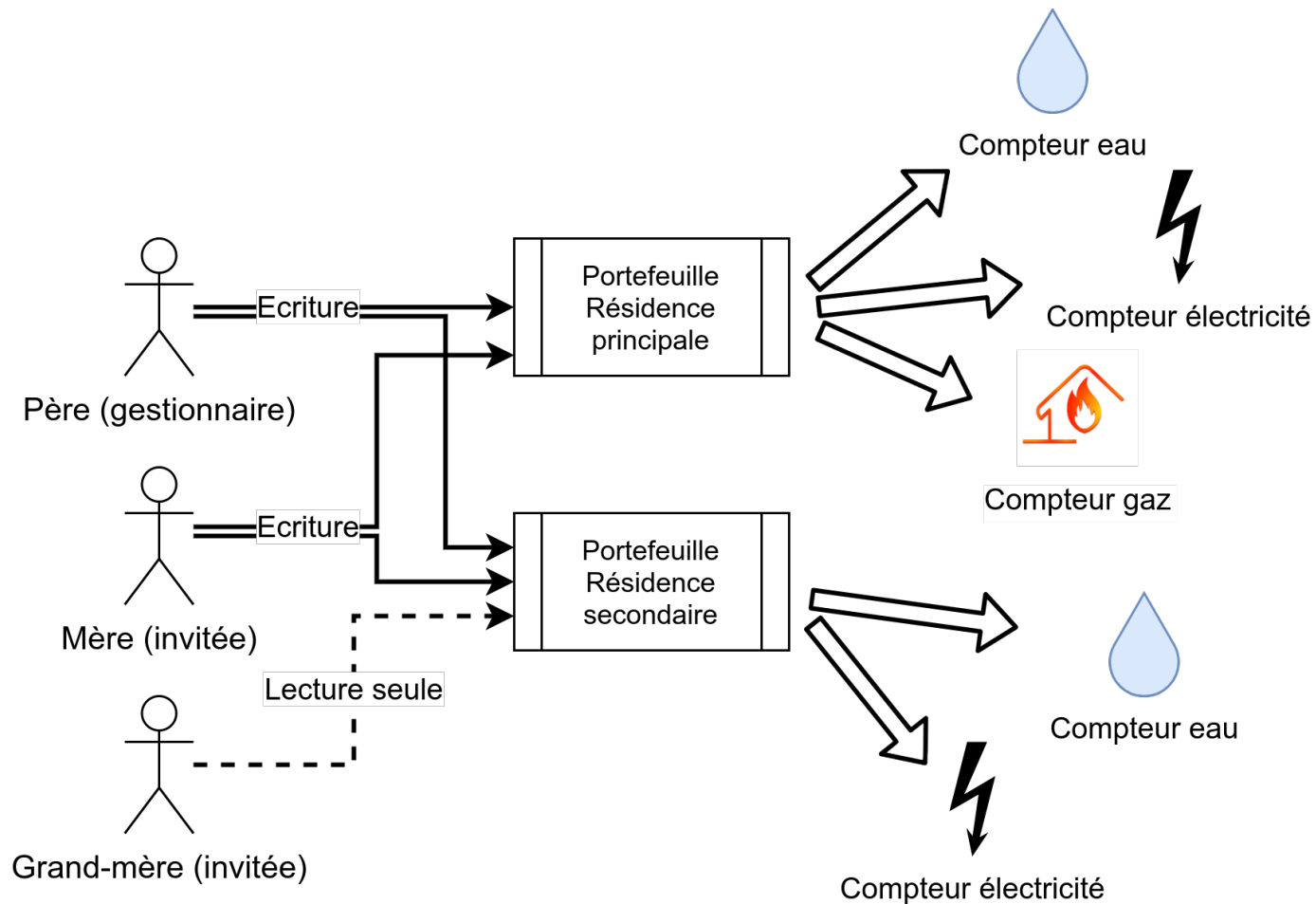
Test-driven development



Copyright 2003-2006 Scott W. Ambler

Projet de génie logiciel

Application web « consommateur » Gestion de portefeuilles énergétiques



Projet de génie logiciel

Exemples de données (format YAML)

```
habitation:  
  habitation-id: h0338384312557  
  adresse:  
    rue: Avenue Maistriau  
    numero: 15  
    batiment: De Vinci  
    local: 2.18  
    ville: Mons  
    code-postal: 7000  
  points-de-fourniture:  
    - point: electricite  
      ean-18: 803429457858963427  
    - point: gaz  
      ean-18: 946430795197304526  
    - point: eau  
      ean-18: 794929267075728442
```


```
consommation:  
  compteur: 803429457858963427  
  releves:  
    - date: 2017-01-05  
      valeur: 7085  
    - date: 2017-01-12  
      valeur: 7123  
    - date: 2017-01-19  
      valeur: 7135  
    - date: 2017-01-26  
      valeur: 7142
```

Projet de génie logiciel

Exemples de données (format YAML)

```
habitation:  
  habitation-id: h0338384312557  
  adresse:  
    rue: Avenue Maistriau  
    numero: 15  
    batiment: De Vinci  
    local: 2.18  
    ville: Mons  
    code-postal: 7000  
  points-de-fourniture:  
    - point: electricite  
      ean-18: 803429457858963427  
    - point: gaz  
      ean-18: 946430795197304526  
    - point: eau  
      ean-18: 794929267075728442
```

```
portefeuille:  
  id: p1589156768969  
  habitation: h0338384312557  
  users:  
    - type: gestionnaire  
      user-id: u0004590088929  
    - type: lecture-seule  
      user-id: u3088953646046  
    - type: lecture-ecriture  
      user-id: u2722500332198  
  points-de-fourniture:  
    - ean-18: 803429457858963427  
    - ean-18: 946430795197304526
```



Projet de génie logiciel

Application web pour les fournisseurs d'énergie

- Gestion de clients et de contrats
- Gestion de compteurs
- Gestion de consommation

Projet de génie logiciel

Exemples de données (format YAML)

```
habitation:  
  habitation-id: h0338384312557  
  adresse:  
    rue: Avenue Maistriau  
    numero: 15  
    batiment: De Vinci  
    local: 2.18  
    ville: Mons  
    code-postal: 7000  
  points-de-fourniture:  
    - point: electricite  
      ean-18: 803429457858963427  
    - point: gaz  
      ean-18: 946430795197304526  
    - point: eau  
      ean-18: 794929267075728442
```

```
fournisseur:  
  nom: Lampiris  
  siege:  
    rue: Saint Laurent  
    numero: 54  
    ville: Liege  
    code-postal: 4000  
  contrats:  
    - client: E08712638  
      numero-contrat: C13871234  
      ean-18: 803429457858963427  
      ean-18: 946430795197304526  
      debut: 2020-07-01
```

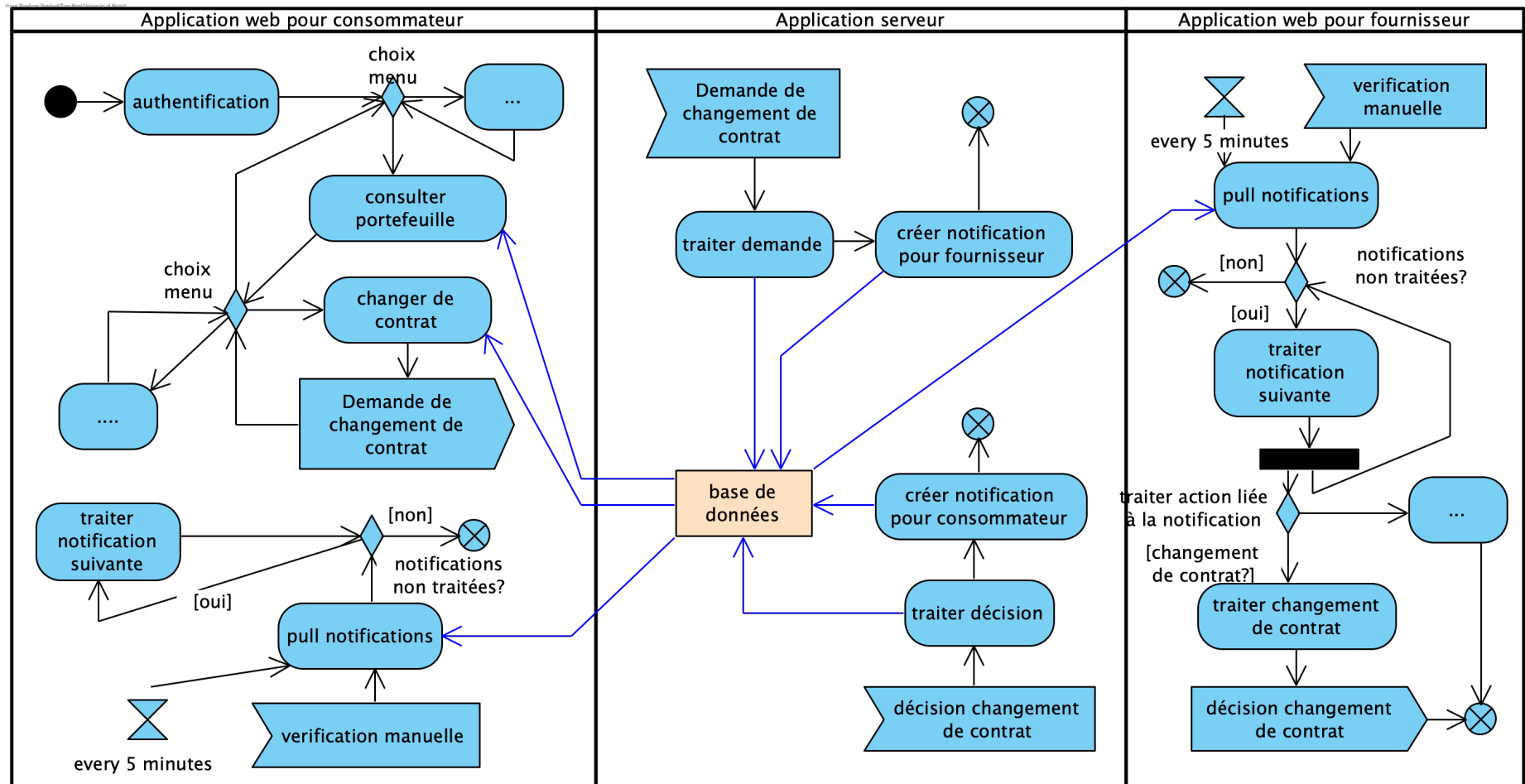


```
fournisseur:  
  nom: SWDE  
  siege:  
    ...  
  contrats:  
    - client: 4569231678  
      numero-contrat: C9987332678  
      ean-18: 794929267075728442  
      debut: 2020-07-01
```



Projet de génie logiciel

Système générique de notifications



Énoncé du projet

Extensions

1. Gestion d'utilisateurs
2. Gestion de portefeuilles agrégés
3. Gestion énergétique pour organisations
4. Analyse statistique de la consommation énergétique
5. Auto-production d'électricité
6. Budget énergétique et rapportage
7. Facturation et paiement d'acomptes
8. Génération de données de consommation et prédiction de consommation future
9. Jeu sérieux: réduction de consommation
10. Application mobile sur Android
11. Application mobile sur iOS

Projet de génie logiciel

Langages et technologies

Phase d'analyse et de conception

- Outil de modélisation:



<https://ap.visual-paradigm.com/university-of-mons>

- UML 2.5
 - Use case diagrams
 - Interaction overview diagrams
 - Class diagrams
 - Sequence diagrams
- Maquette de l'UI

Projet de génie logiciel

Langages et technologies

Phase d'analyse et de conception

- Concentrez vos efforts de modélisation sur les comportements *intéressants* et *complexes* de l'application à réaliser plutôt que d'écrire 500 scénarios triviaux qui ne contribuent pas beaucoup à la compréhension du problème...
- N'oubliez pas de modéliser la fonctionnalité de base ET la fonctionnalité des extensions!

Projet de génie logiciel

Langages et technologies

Phase d'analyse et de conception

- Le design du *schema de la base de données* et de *l'API REST* est **essentiel**, car elles seront utilisées par toutes les applications clients.
- Il y aura une seule API REST pour accéder au serveur, et une seule base de données.
- L'API et la base de données seront communes pour la fonctionnalité de base et toutes les extensions. Il est donc important de les réaliser en groupe.

Projet de génie logiciel

Langages et technologies

Phase d'analyse et de conception

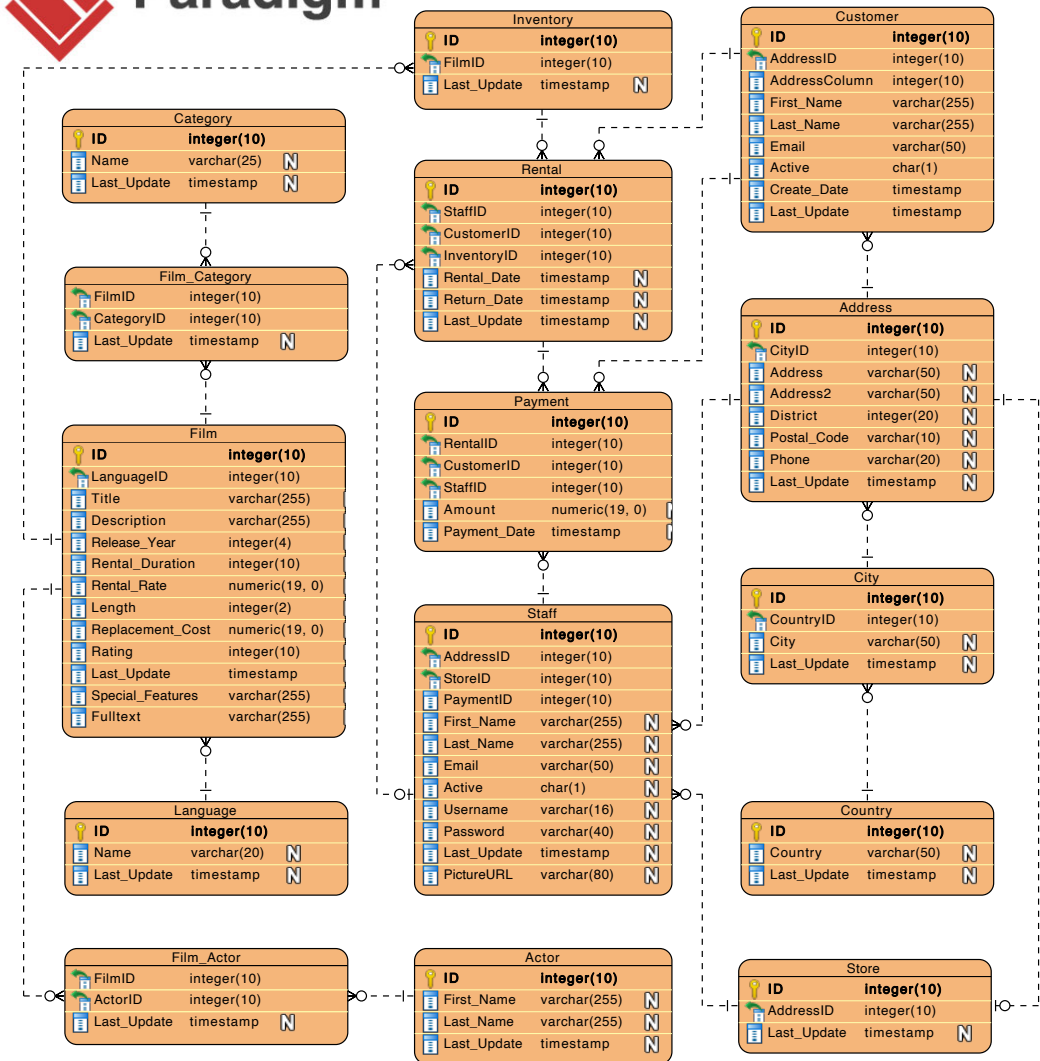


- Concentrez vos efforts de modélisation sur les comportements *intéressants* et *complexes* de l'application à réaliser plutôt que d'écrire 500 scénarios triviaux qui ne contribuent pas beaucoup à la compréhension du problème...
- N'oubliez pas de modéliser la fonctionnalité de base ET la fonctionnalité des extensions!

Projet de génie logiciel

Langages et technologies

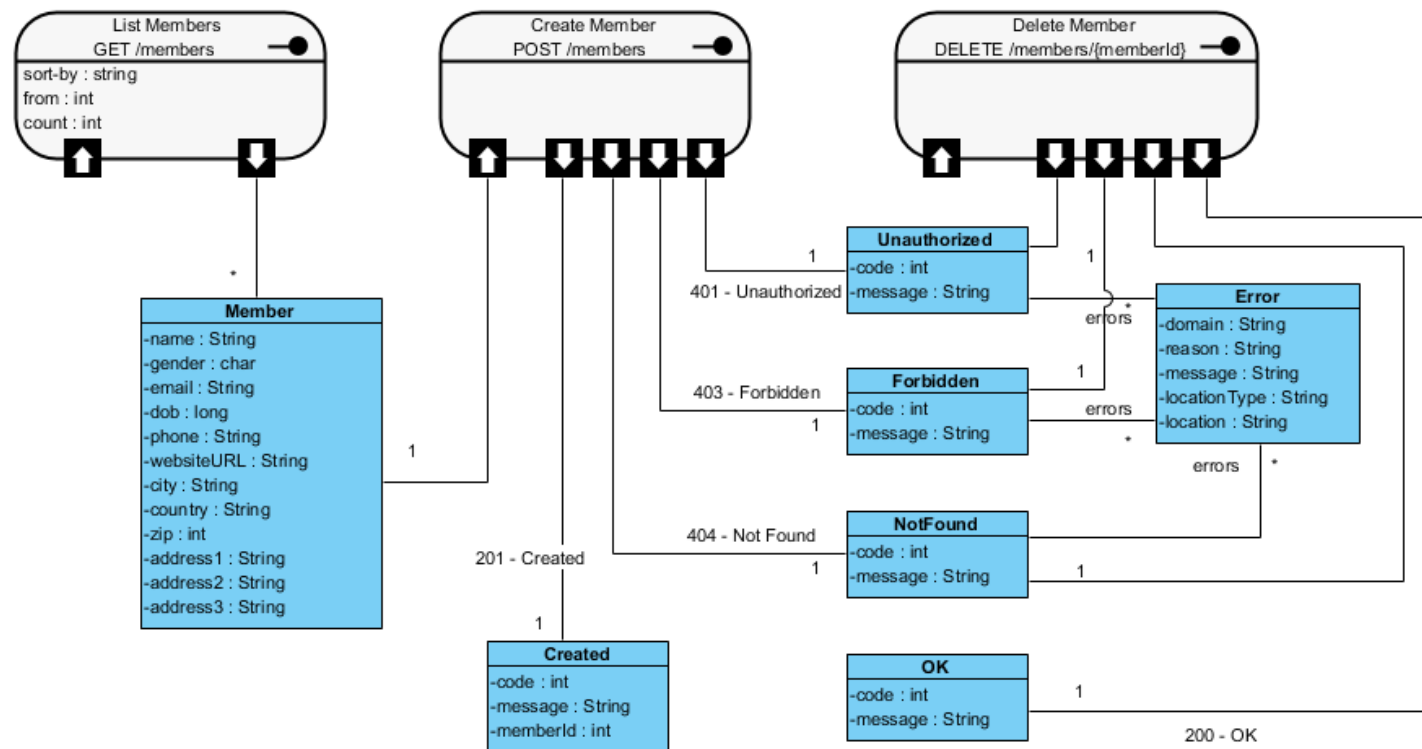
Diagrammes d'entité-relation (ERD)



Projet de génie logiciel

Langages et technologies

Visual design of REST API



Projet de génie logiciel

Langages et technologies

Technologies

- Gestion de versions:



- Moteur de production:



- Outil de développement:



Projet de génie logiciel

Langages et technologies

Pour la phase de développement

- Coté serveur:
 - Java 17 (LTS)
 - JavaDoc
 - JUnit 5 + mocking (e.g. Mockito)
 - Base de données: au choix (relationnelle ou NoSQL)
- Librairie/framework REST: au choix
- Technologies JavaScript: au choix

Langages et technologies

JavaDoc

Génération automatique de la documentation HTML à partir des commentaires dans le code Java

@author – who wrote this code

@version – when was it changed

@param – describe method parameters

@return – describe method return values

@throws – describe exceptions thrown

@see – link to other, related items (e.g. “See also...”)

@since – describe when code was introduced (e.g. API Level)

@deprecated - describe deprecated item and what alternative to use instead

Langages et technologies

JavaDoc

Example

```
/**
 * Activity for loading
 * layout resources
 *
 * This activity is used to ...
 *
 * @author LED
 * @version 2010.1105
 * @since 1.0
 */
public class LayoutActivity
extends Activity {
```

Package [Class](#) [Use](#) [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#)
[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

com.androidbook.layoutft

Class LayoutActivity

java.lang.Object
└ Activity
└ com.androidbook.layoutft.LayoutActivity

```
public class LayoutActivity
extends Activity
```

Activity for loading layout resources This activity is used to display different layout resources for a tutorial on user interface design.

Since:
1.0

Version:
2010.1105

Author:
LED

Langages et technologies

JavaDoc

Example

```
/**
 * Method that adds two integers together
 *
 * @param a The first integer to add
 * @param b The second integer to add
 * @return The resulting sum of a and b
 */
public int addIntegers(int a, int b) {
    return (a+b);
}
```

```
/** * This method simply throws an Exception if the incoming
parameter a is not a positive number, just for fun.
*
```

```
* @param a Whether or not to throw an exception
* @throws Exception
*/
public void throwException(boolean shouldThrow)
    throws Exception {
    if (shouldThrow == true) {
        throw new Exception();
    }
}
```

addIntegers

```
public int addIntegers(int a,
                       int b)
```

Method that adds two integers together

Parameters:

a - The first integer to add
b - The second integer to add

Returns:

The resulting sum of a and b

throwException

```
public void throwException(boolean shouldThrow)
    throws java.lang.Exception
```

This method simply throws an Exception if the incoming parameter a is not a positive number, just for fun.

Parameters:

a - Whether or not to throw an exception

Throws:

java.lang.Exception

Langages et technologies

Unit Testing + Mocking

Unit Testing



Framework automatique
de tests unitaires

Mocking



Utiliser des “mock
objects”, pour simuler le
comportement d’objets
ou modules complexes

- Interface graphique
- Base de données
- Serveur web
- Calculs complexes

Langages et technologies

```
import java.util.List;
import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class MockitoMockMethodExample {

    @SuppressWarnings("unchecked")
    @Test
    public void test() {
        // using Mockito.mock() method
        List<String> mockList = mock(List.class);
        when(mockList.size()).thenReturn(5);
        assertTrue(mockList.size()==5);
    }
}
```



Langages et technologies

```
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

public class MockitoMockAnnotationExample {
    @Mock
    List<String> mockList;

    @BeforeEach
    public void setup() {
        MockitoAnnotations.initMocks(this);
    }

    @SuppressWarnings("unchecked")
    @Test
    public void test() {
        when(mockList.get(0)).thenReturn("MockitoExample");
        assertEquals("MockitoExample", mockList.get(0));
    }
}
```



Langages et technologies

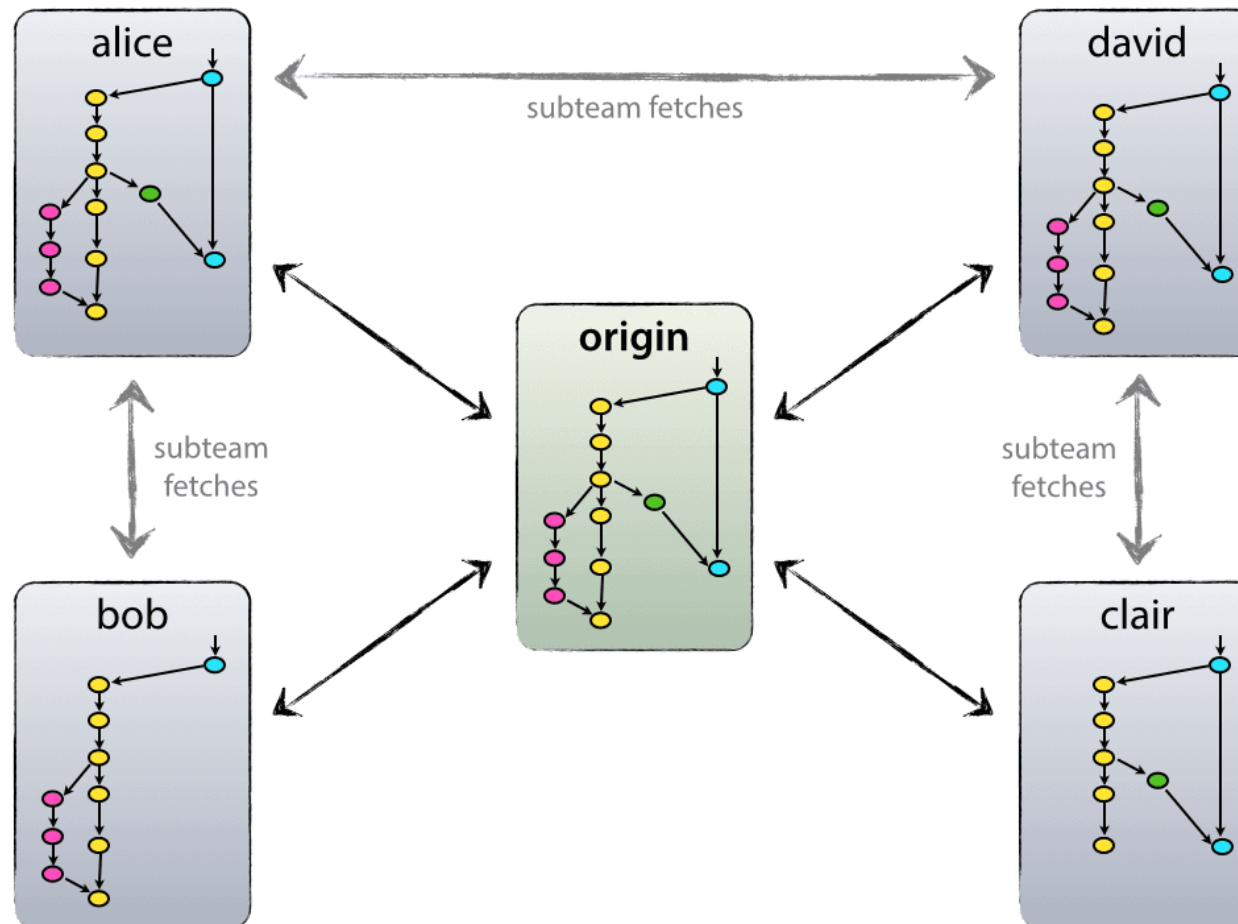
Git

Tutoriel git en ligne

https://learngitbranching.js.org/?locale=fr_FR

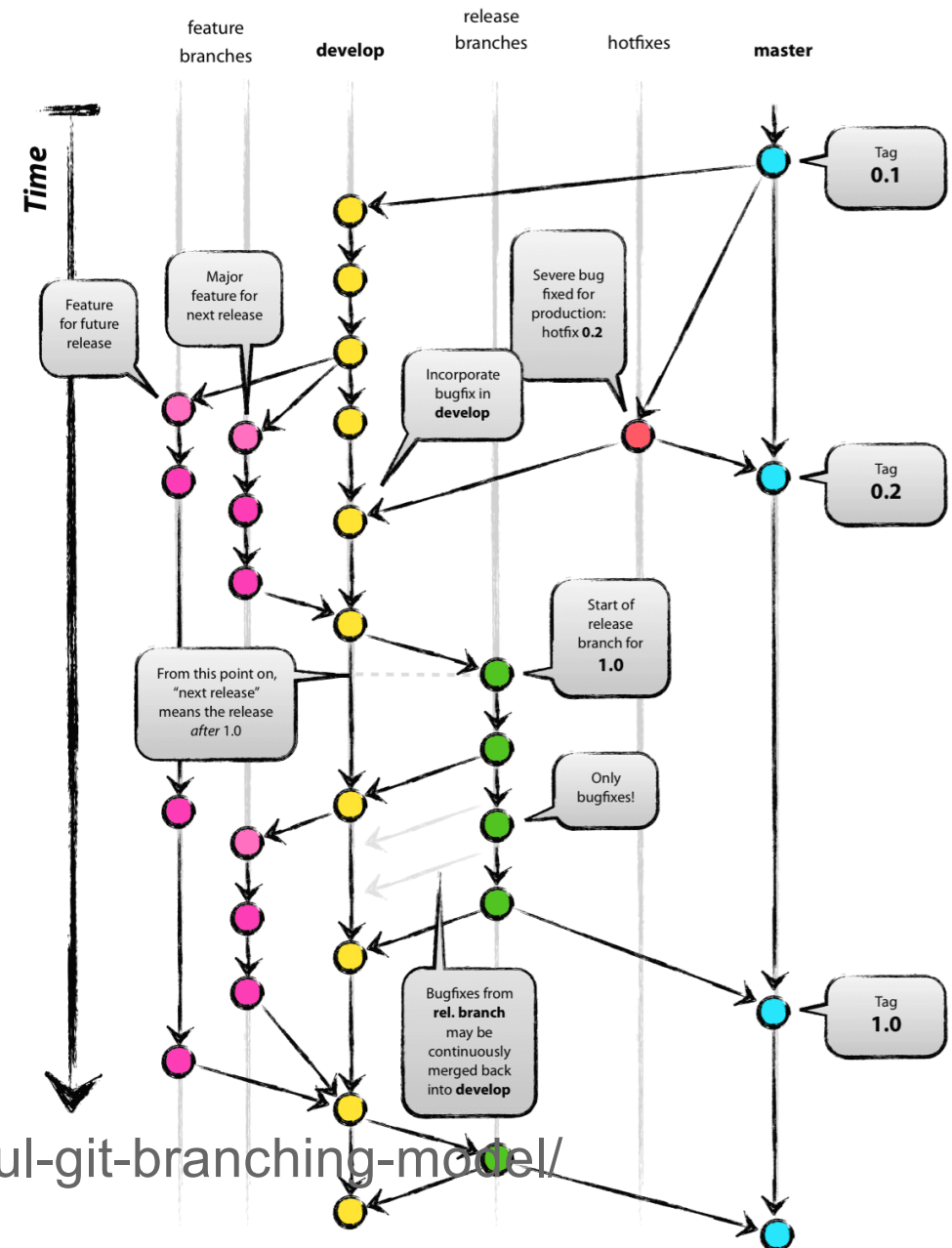
Langages et technologies

Git: Distributed version control
<https://git-scm.com>



Langages et technologies

Git: Example of a useful branching model



<https://nvie.com/posts/a-successful-git-branching-model/>

Langages et technologies

Exemple script d'automatisation build.gradle

```
plugins {  
    // java plugin to add support for Java  
    id 'java'  
    // application plugin to add support for building a CLI application.  
    id 'application'  
    // javafx plugin  
    id 'org.openjfx.javafxplugin' version '0.0.8'  
}  
repositories {  
    mavenCentral()  
}  
javafx {  
    modules = [ 'javafx.controls', 'javafx.fxml' ]  
}  
dependencies {  
    // Junit 5 Jupiter for testing.  
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.5.2'  
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.5.2'  
    // TestFX to test the GUI.  
    testImplementation "org.testfx:testfx-junit5:4.0.16-alpha"  
}  
application {  
    // Define the main class for the application.  
    mainClass = 'gui.Main'  
}  
...
```



Langages et technologies

```
...

test {
    // Use junit platform for unit tests
    useJUnitPlatform()
}

jar.baseName('SimpleUI')

mainClassName='gui.Main'

jar {
    manifest {
        attributes 'Main-Class': 'gui.Main'
    }

    from {
        // Avoids conflicts from each jar of javafx having this exact same file
        exclude "**/module-info.class"
        // Add all the dependencies in the jar.
        configurations.compileClasspath.collect {zipTree(it)}
    }
}
```