



Projet

Jeu sérieux : Placement de navires sur les quais d'un port de commerce

Spécification technique

Groupe 2 → Gary Hébert
Jérémy Courel
Mounir El Mtakham
Alaaeddine Kheribeche

Clients → M. Eric Sanlaville
M. Stefan Balev

Sommaire

1. Choix techniques.....	3
1.1. Type de client (web/bureau).....	3
1.2. Langage de programmation.....	3
1.3. Détails sur l'IHM.....	4
1.4. Stockage des données.....	4
1.4.1. Local.....	4
1.4.2. Stockage distant.....	4
1.5. Architecture logicielle.....	5
2. Diagramme de classes.....	7
3. Diagramme de déploiement.....	7
4. Création de la carte du port du havre.....	8
5. Génération des fichiers d'instance.....	9
6. Format de stockage des préférences.....	10
7. Format de stockage des scores en JSON.....	11
8. Modèle conceptuel de donnée pour les scores en ligne.....	11

1. Choix techniques

Avant de construire un projet, et après avoir traité la partie fonctionnelle, nous devons prendre des décisions sur les technologies à utiliser lors de la phase de réalisation de celui-ci.

Nous avons décidé de ces technologies par étape et selon nos connaissances actuelles, ceci dû au temps limité pour la réalisation du projet.





1.1. Type de client (web/bureau)

Le client sera de type « desktop ». En effet, un projet web aurait été une tout aussi bonne solution à l'exception de deux points :

- Cela nécessite une connexion constante à internet pour l'utiliser ;
- Cela impliquait d'utiliser des technologies que nous ne maîtrisons pas (webGL, API google maps, JEE) et donc un risque que le projet n'aboutisse pas à une version fonctionnelle.

1.2. Langage de programmation

Nous avons hésité entre plusieurs langages de programmation et voici ce qui nous a poussé à choisir le Java :

				
<i>Orienté objet</i>	✓	✗	✗	✓
<i>Connaissance</i>	☆☆☆	☆	☆	☆☆
<i>3D native</i>	✓	✗	✗	✗
<i>Bureau / Web</i>	✓ / ✓	✓ / ✗	✓ / ✗	✗ / ✓

Le java est un langage de programmation fortement typé, orienté objet et permettant de créer à la fois des applications bureau (JSE, JEE) et des applications web (JEE, applet).

Il permet de plus, dans sa version standard, de gérer n'importe quel type de base de données (via PDO pour le SQL et des objets particuliers pour les bases NoSQL), d'afficher en 2D (AWT/Swing) et en 3D (java3D).

Java permet donc, dans sa version standard, de réaliser le projet quasiment entièrement (sans l'utilisation de bibliothèque externe).

1.3. Détails sur l'IHM

Voici quelques détails sur la réalisation de l'IHM :

- Réalisée en Swing ;
- Dessin 2D avec l'objet « Graphics » de Java (pour tracer la carte du port du Havre et interagir avec celle-ci) ;
- 3D réalisée via Java3D car c'est intégré nativement dans Java (c'est la raison pour laquelle nous avons opté pour cette technologie au lieu d'utiliser « JOGL → Java for OpenGL »).

1.4. Stockage des données

1.4.1. Local

Certaines données seront stockées sur la machine locale. Ces données sont directement en rapport avec l'utilisateur. En effet, le stockage des préférences dans les options de l'application n'a aucune raison d'être sur un serveur distant. Une version des scores sera stockée en local pour permettre aux utilisateurs d'une même machine de comparer leurs scores.

Pour ces données, nous utiliserons une base de données NoSQL sous forme de fichier JSON, permettant de stocker de manière propre et légère (les fichiers sous format JSON ont une taille très réduite) et sans l'utilisation d'un système de gestion de base de données (SGBD) particulier.

1.4.2. Stockage distant

Les principaux SGBD sur le marché offrent un stockage efficace, la décision ne s'est donc pas faite sur un critère objectif. Notre principal critère a été notre connaissance de ce SGBD, le fait qu'il offre ou non une version complète gratuite et la probabilité de l'utiliser dans un environnement professionnel.

				
Gratuit	✓	✓	✗	✗
Connaissance	☆☆☆	☆☆☆	☆☆	☆
Utilisation professionnelle	✓	✗	✓	✓

Nous avons donc opté pour PostgreSQL, SGBD complet, gratuit et professionnel.

1.5. Architecture logicielle

Nous avons choisis une architecture séparé en 4 « package » qui sont :

- Accès au données

Ce « package » contiendra tous les accès aux fichiers (fichier de score local, fichier de paramètre du jeu) ainsi que les accès à la base de données.

- Métier

Ce « package » contiendra le traitement des événements lié à la couche présentation, ainsi que le traitement des informations récupérées dans des fichiers ou dans la base de données.

- Présentation

Ce « package » contiendra tous ce qui est interface utilisateur, tous ce que celui-ci visualisera sera créé dans ce « package ».

- Modèle

Ce « package » contiendra quant à lui tous les éléments qui nous permettront de manipuler nos données, tel que la classe Navire, la classe Quai, ... Ce « package » décrira donc nos données manipulées.

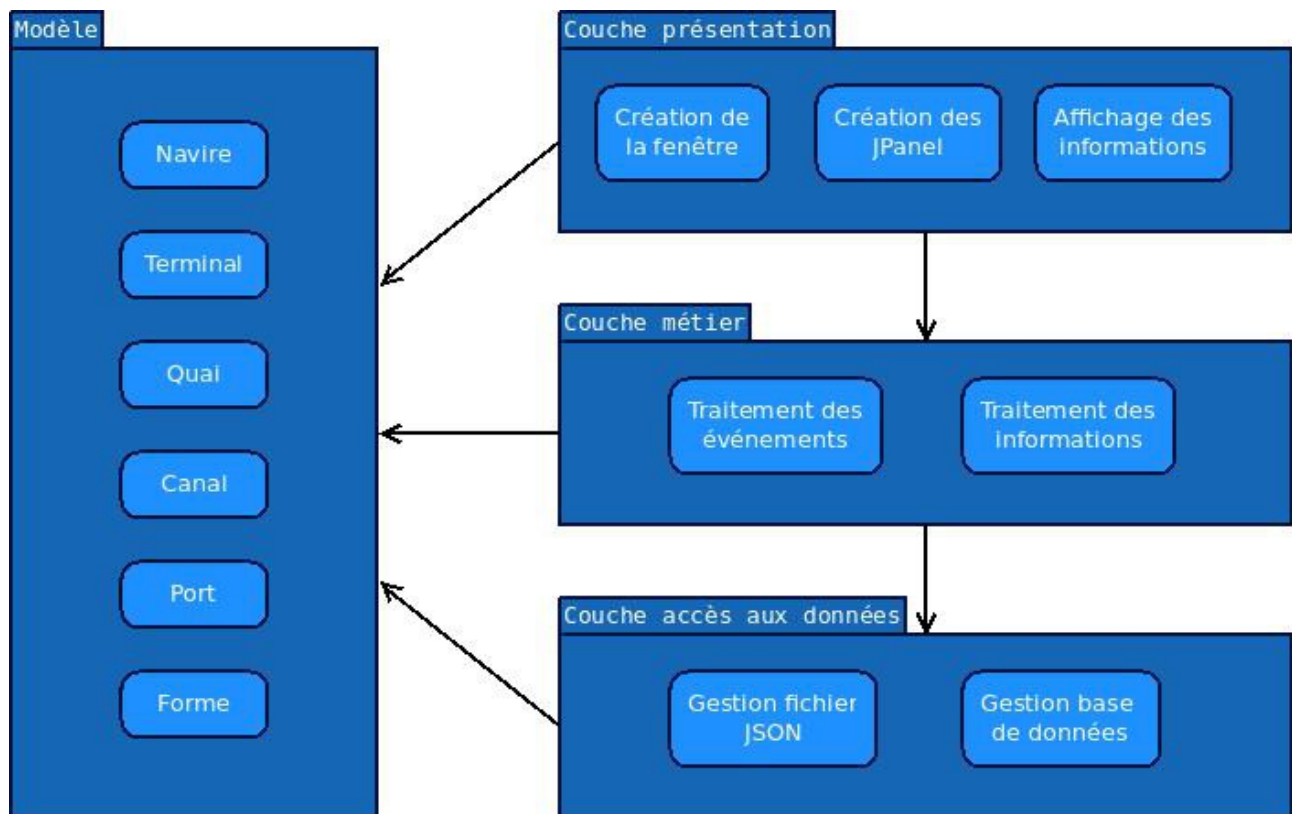
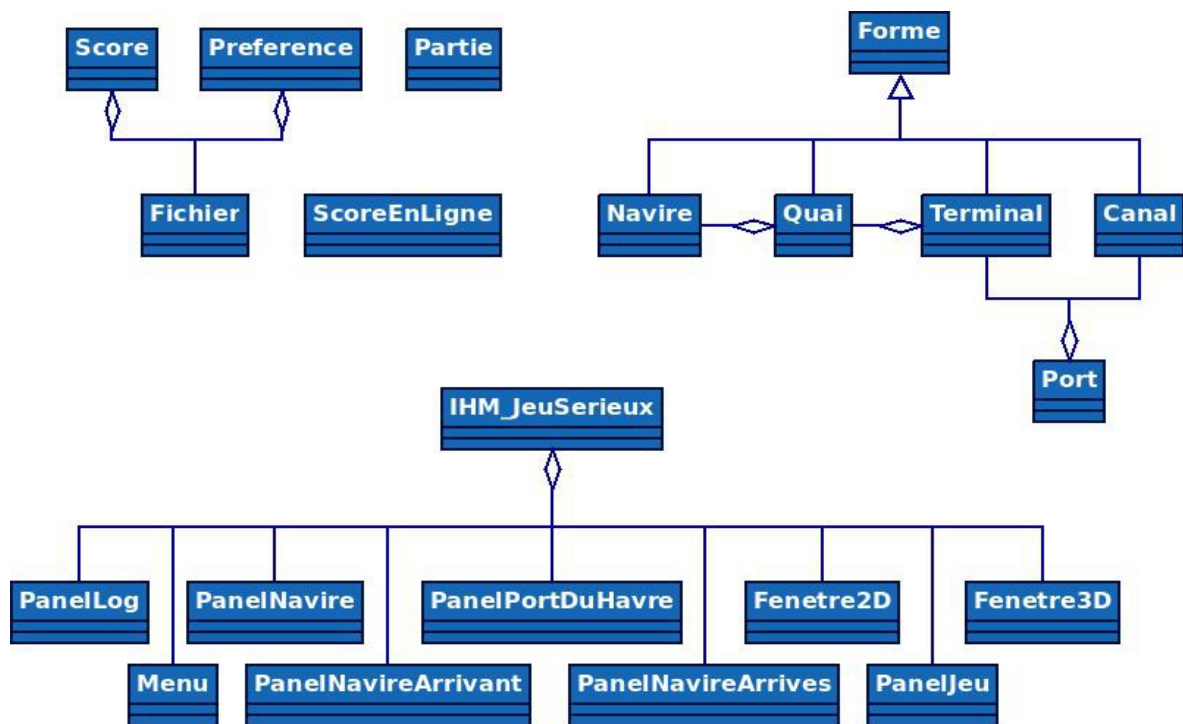


Illustration 1: Diagramme illustrant l'architecture logicielle choisie

2. Diagramme de classes



3. Diagramme de déploiement

Le [diagramme de déploiement](#) permet de voir les interactions entre les différents matériaux dont le jeu aura besoin. Dans ce diagramme, 2 nœuds sont identifiés :

- **PC** (il contient 3 composants)
 - Base de données noSQL ;
 - Jeu sérieux ;
 - Navigateur.
- **Serveur** (il contient 1 composant)
 - Base de données SQL.

Ce diagramme indique qu'il y aura des interactions entre :

- La base de données noSQL et le jeu sérieux (pour le stockage en local des scores et des

préférences de l'utilisateur) ;

- Le jeu sérieux et la base de données SQL (pour le stockage en ligne des scores et les compte utilisateurs associés à ces scores) ;
- Le navigateur et la base de données SQL (pour la connexion de l'utilisateur et la consultation des scores en ligne).

Les interactions entre l'ordinateur de l'utilisateur et le serveur sur lequel sera stocké la base de données SQL se feront en TCP/IP.

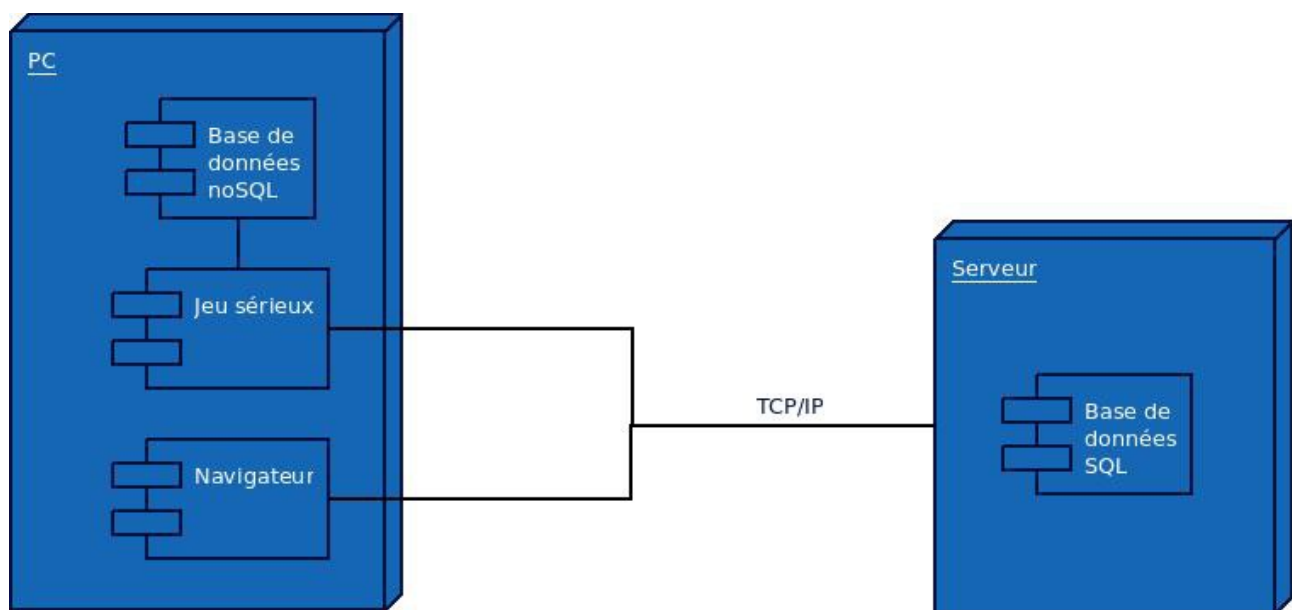


Illustration 2: Diagramme de déploiement

4. Création de la carte du port du havre

Pour avoir une carte du port du havre modifiable dans notre jeu, nous allons récupérer un fichier XML contenant tous les points à tracer pour l'obtenir, ce fichier sera récupéré sur le site openstreetmap (à cette adresse → <http://www.openstreetmap.org>), pour obtenir les points de ce que l'on souhaite il suffit de sélectionner une partie de la carte et ensuite d'exporter cette sélection.

Après avoir récupéré ce fichier, nous allons :

- Extraire les informations qui nous intéressent du fichier XML ;
- Créer une version simplifiée du fichier au format JSON ;

- Dessiner le port du havre grâce au fichier JSON (via l'objet « Graphics » de java).

Notre objectif est d'avoir une carte sur laquelle on puisse afficher :

- Les navires assignés à leur quai ;
- Les quais disponibles ;
- Les quais indisponibles ;
- Le chemin emprunté par les navires.

5. Génération des fichiers d'instance

Pour que le jeu ne soit pas redondant il faut que celui-ci soit capable de générer aléatoirement ses fichiers en entrée, pour effectuer cela on passera par ces étapes :

- Étude du transit portuaire du Havre pour créer des parties réaliste (et proposer plusieurs niveaux de difficulté) ;
- Génération aléatoire de chaque navire (type aléatoire, taille aléatoire, ...), il faut que les types de navires ne soient pas tous les même pour un même temps donc ça sera de l'aléatoire orienté pour certains paramètres ;
- Sauvegarde de ces données au format JSON sous cette forme :

```
{
  "temps": {
    "navires": [
      {
        "nom": "nomDuNavire1",
        "type": "typeDuNavire1",
        "longueur": "taille1",
        "dateArrivee": "date1",
        "tempsTraitement": "temps1"
      },
      {
        "nom": "nomDuNavire2",
        "type": "typeDuNavire2",
        "longueur": "taille2",
        "dateArrivee": "date2",
        "tempsTraitement": "temps2"
      }
    ]
  },
  "temps": {
    "navires": [
      {
        "nom": "nomDuNavire3",
        "type": "typeDuNavire3",
        "longueur": "taille3",
        "dateArrivee": "date3",
        "tempsTraitement": "temps3"
      },
      {
        "...": "..."
      }
    ]
  }
}
```

6. Format de stockage des préférences

Les préférences de l'utilisateur seront stockées au format JSON, voici les différentes informations à enregistrer :

- Pseudo utilisé par l'utilisateur ;
- Compte utilisé par l'utilisateur ;
- Difficulté utilisé
- ??

7. Format de stockage des scores en JSON

Les scores seront stockés au format JSON sous ce format :

```
{
  "joueur1": {
    "score": "4000",
    "difficulte": "facile",
    "durée": "500h"
  },
  "joueur2": {
    "score": "1000",
    "difficulte": "difficile",
    "durée": "1000h"
  },
  "...": {
    }
}
```

les principales informations à stocker sont :

- le score de la partie ;
- la difficulté de la partie ;
- la durée de la partie ;
- ??

8. Modèle conceptuel de donnée pour les scores en ligne

Pour que les utilisateurs puissent comparer leurs scores avec ceux d'autre personne, il faut une base de données en ligne pour cela, voici le modèle conceptuel de données de cette BDD :

