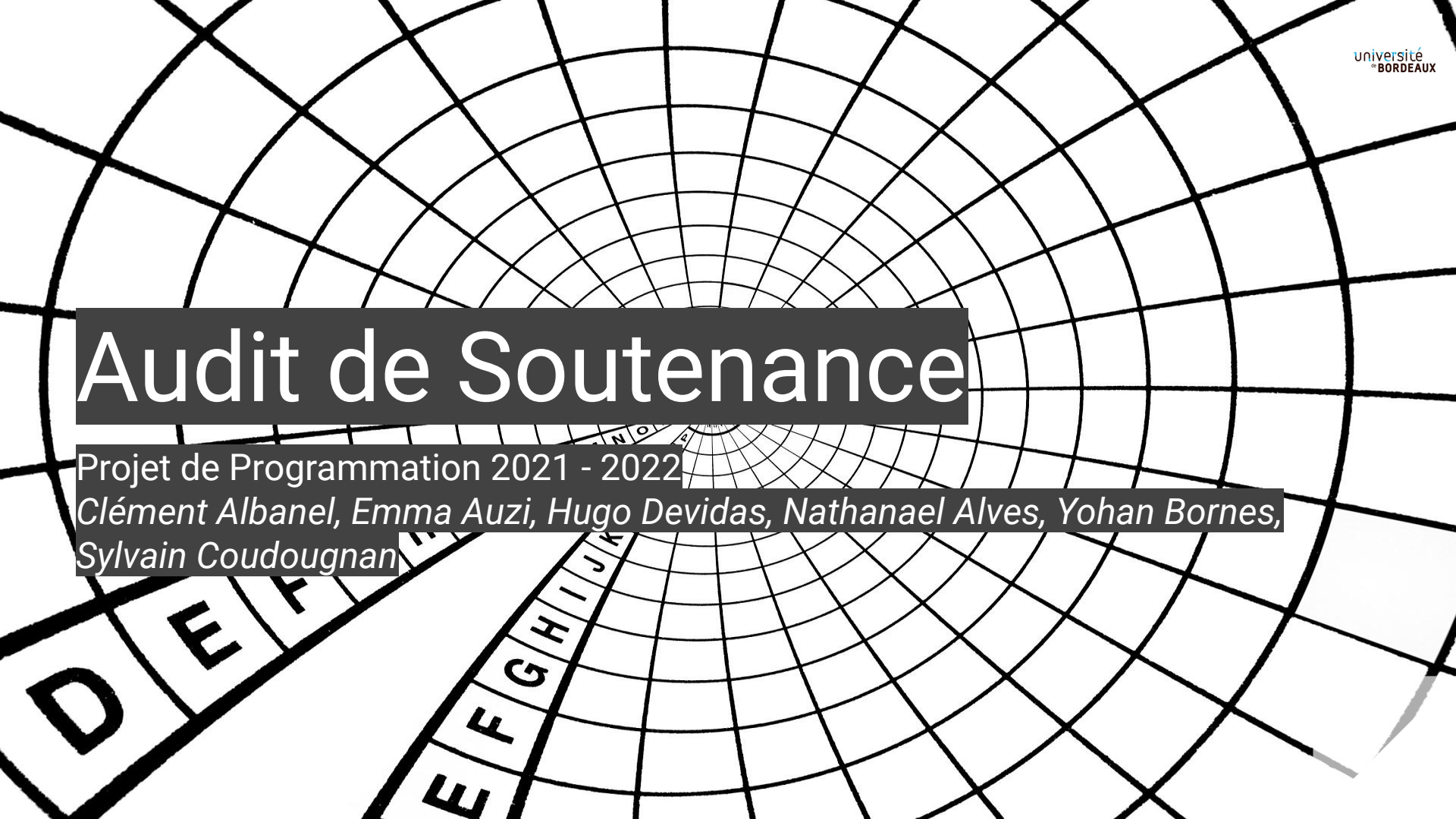
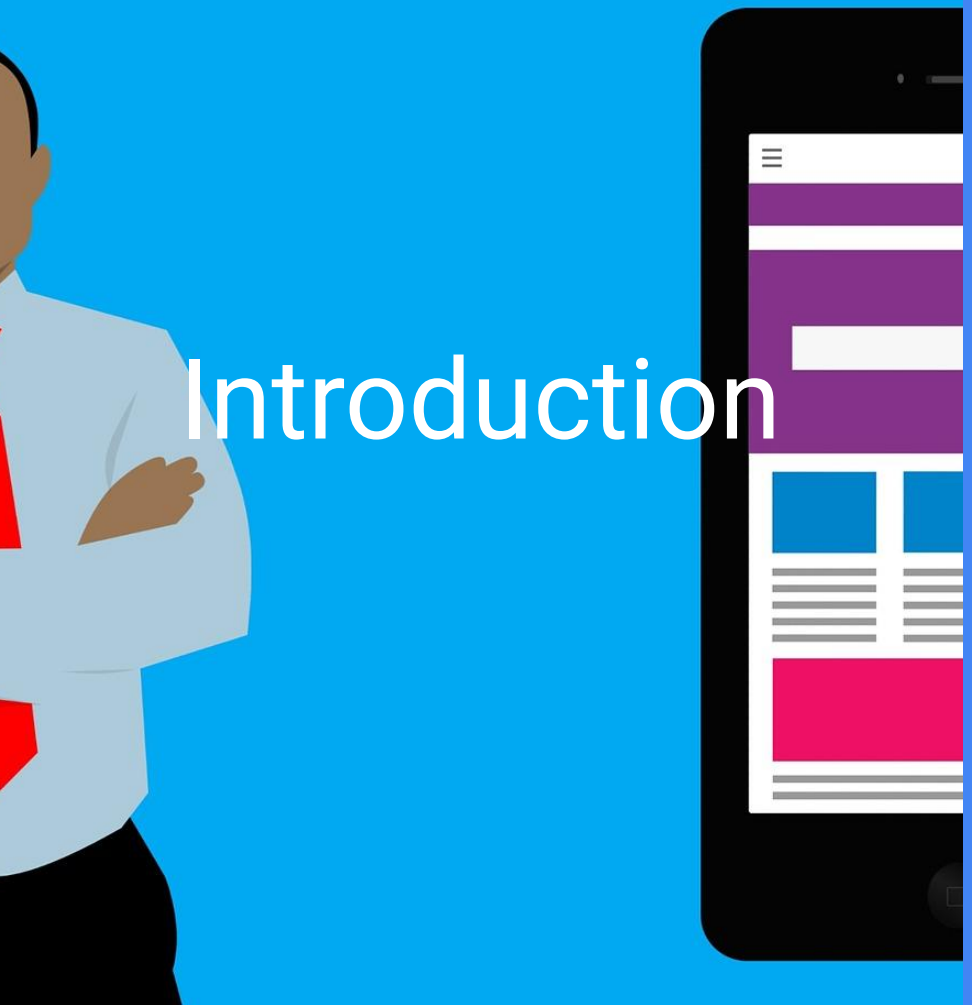


Audit de Soutenance

Projet de Programmation 2021 - 2022

Clément Albanel, Emma Auzi, Hugo Devidas, Nathanael Alves, Yohan Bornes, Sylvain Coudougnan



Introduction

- Création d'un programme capable de **générer** automatiquement des grilles de mots croisés
- L'utilisateur doit pouvoir entrer en **interaction** avec le programme (i.e y jouer), paramètrer son jeu (niveaux de difficultés variables)
- Utilisations **d'algorithmes** pré-existants en poussant l'efficacité algorithmique au maximum

Les étapes du projet

Janvier - Février 2022

Analyse de **l'existant**
Rédaction des **besoins** / cahier des charges techniques

Avril 2022

Audit de **présentation**
Livraison du projet au client

Jan. | Févr. | Mars | Avr. | Mai | Juin | Juil. | Août | Sept. | Oct. | Nov. | Déc

Mars 2022

Architecture logicielle
Implémentation

Synthèse de l'existant

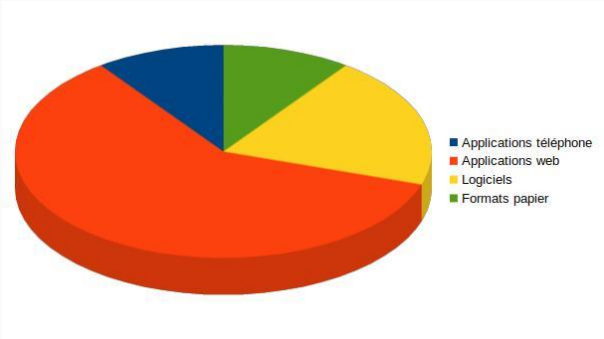


Figure 1 : Proportion des sources de nos recherches sur l'existant

- Web : **60 %** des applications en corrélation avec notre projet → **facilité** d'utilisation pour l'utilisateur
- Notre choix s'est néanmoins orienté vers une application **logicielle** → abstraction des "**conventions du web**" (requêtes HTTP …)
- Besoins presque similaires aux nôtres :
  - Public visé : joueurs de mots croisés tous **niveaux** (niveaux de difficultés variables)
  - Interface graphique simple et **efficace**
  - **Rapidité** de génération

Notre positionnement par rapport à l'existant

Nos valeurs

- Allier **simplicité**, rapidité algorithmique et **fluidité**
- Proposer un mode "**debug**" pour les développeurs (mode terminal) → très peu proposé sur l'existant étudié
- Logiciel "**Open-Source**" → l'utilisateur doit pouvoir connaître les techniques algorithmiques utilisées et itérer dessus s'il le souhaite

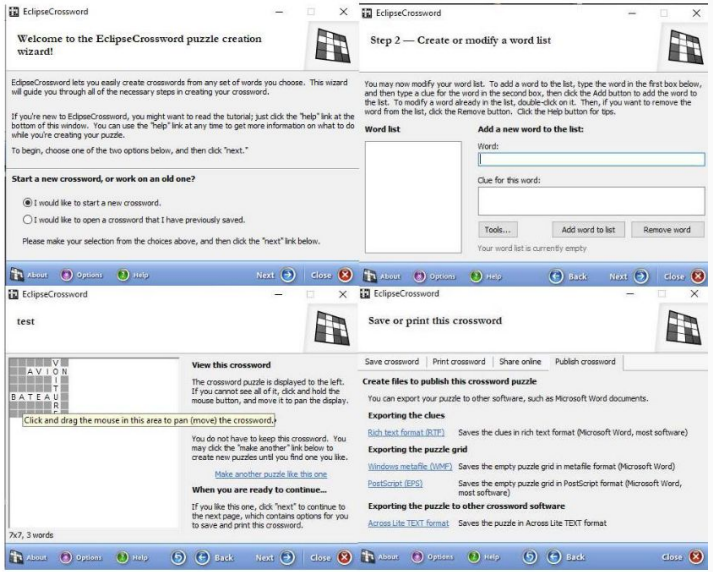


Figure 2 : une interface existante, "surchargée et complexe"

I/ Les besoins du projet

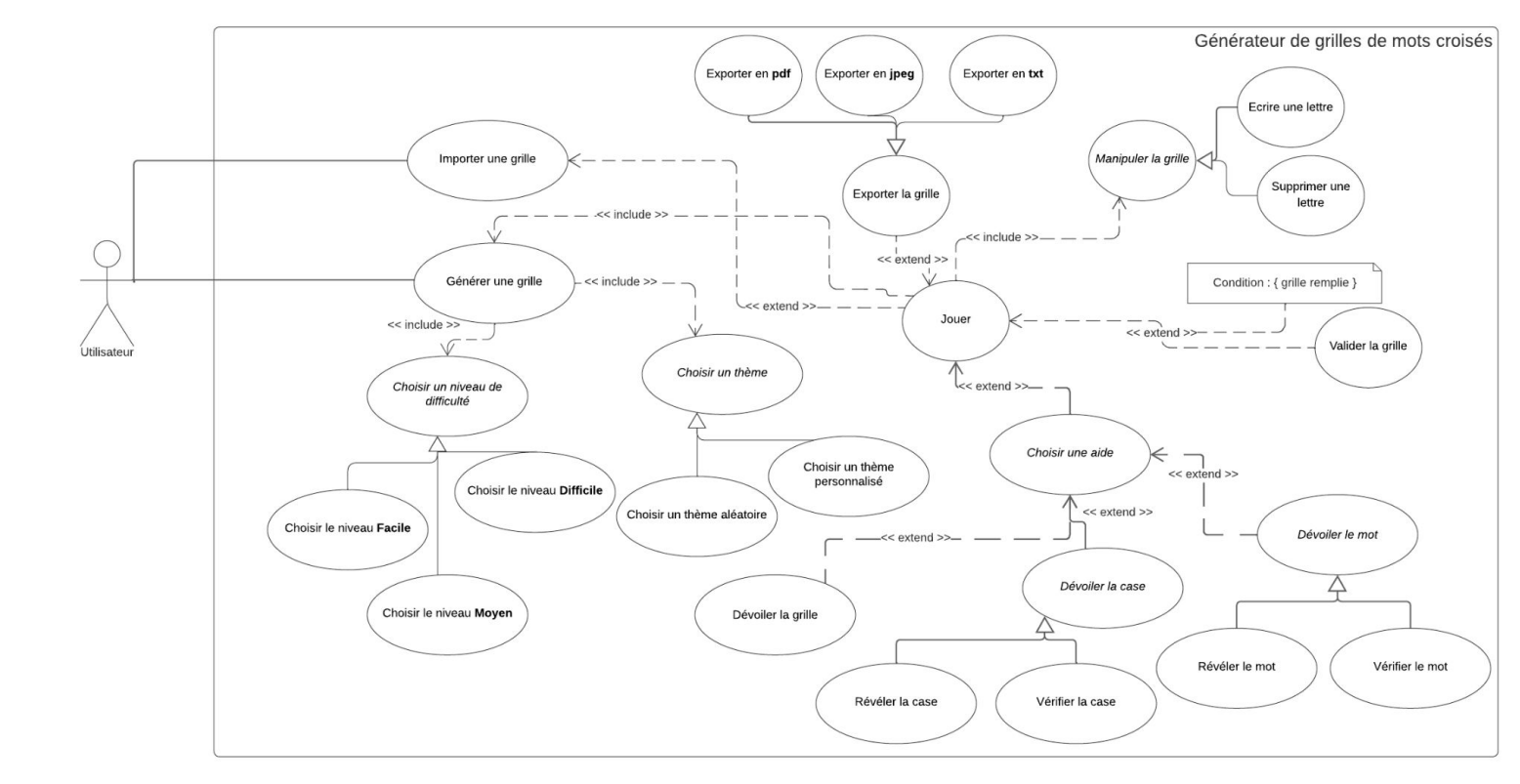


Figure 3 : Diagramme des cas d'utilisation

I/ Les besoins du projet

*Des besoins **fonctionnels** (principaux)*

- **Générer** une grille en appliquant un niveau de difficulté et y jouer
- Bénéficier d'aides utilisateurs (révéler mot/case …)
- Exporter / importer une grille

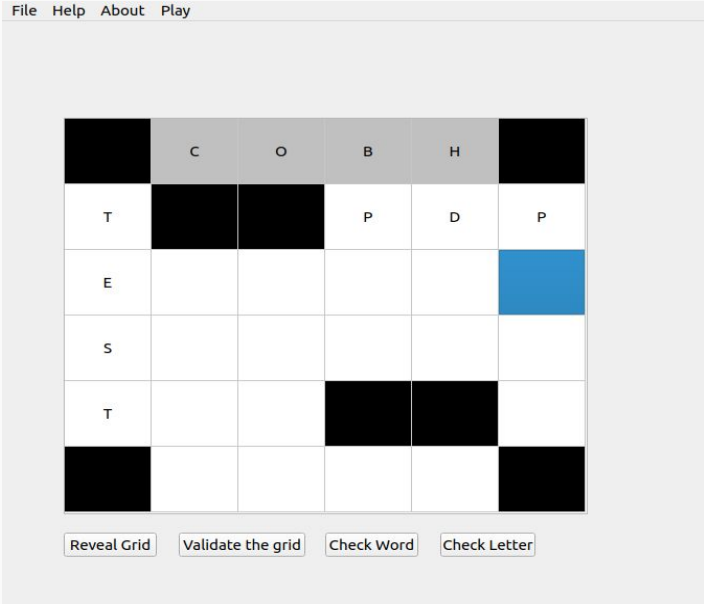


Figure 4 : l'interface actuelle du projet

I/ Les besoins du projet

*Des besoins **non fonctionnels** (principaux)*

- **Comportementaux :** la grille est générée dans un temps très court (de l'ordre de la seconde)
- **Fiabilité & simplicité :** le code produit est robuste et sûr. L'application sera aisée d'utilisation.
- **Interopérabilité :** l'application est en mesure d'interagir avec l'explorateur de l'utilisateur afin d'importer/exporter une grille

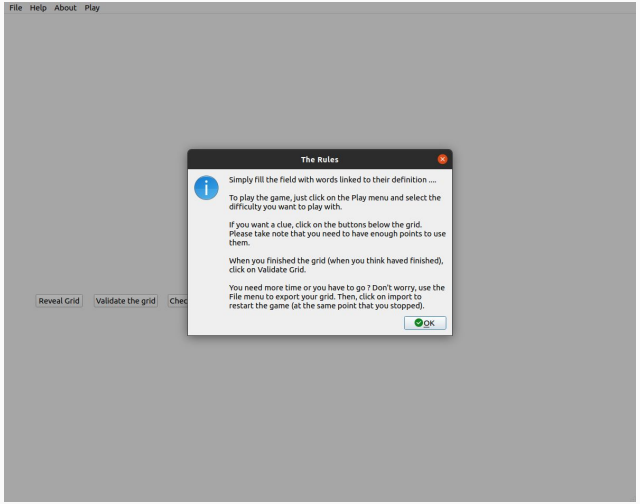
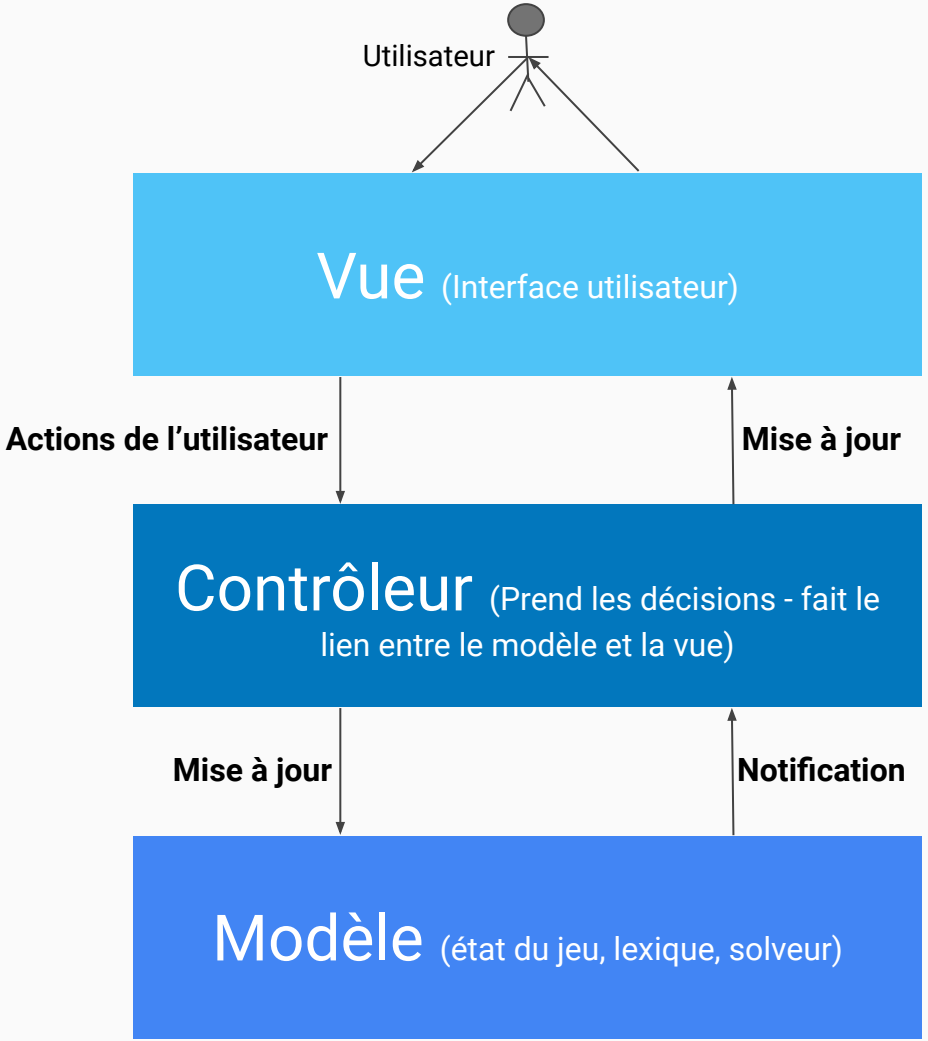


Figure 5 : l'utilisateur peut à tout moment consulter un "mémento" via le menu "About"

II/ L'architecture du logiciel

- Utilisation du motif d'architecture **MVC** (Modèle Vue Contrôleur) → découpage **fonctionnel**



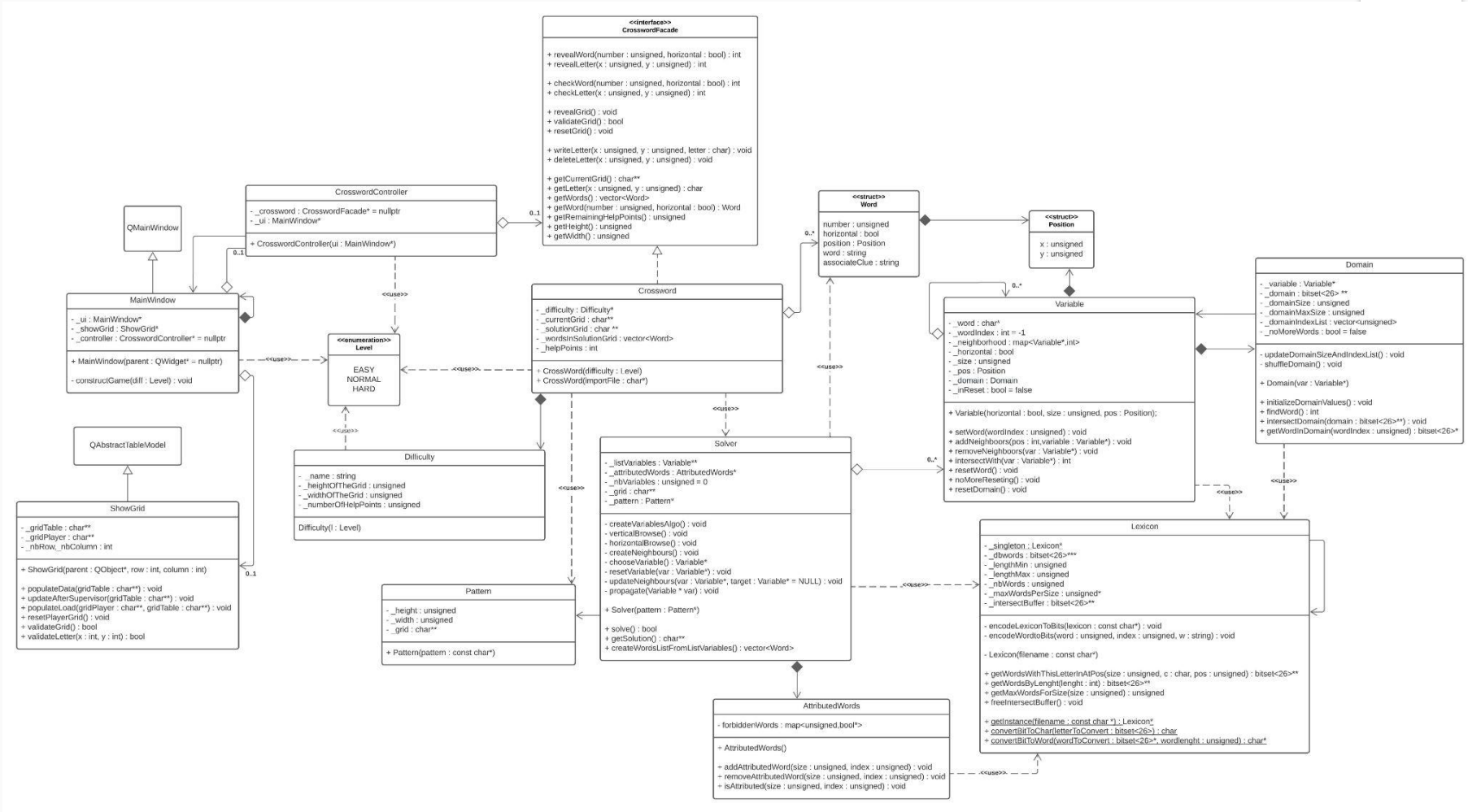


Figure 6 : diagramme de classes UML en l'état de conception actuelle

II/ L'architecture du logiciel

- Limiter les dépendances
- Chacune des classes ont un rôle précis
- Utilisation de certains design patterns

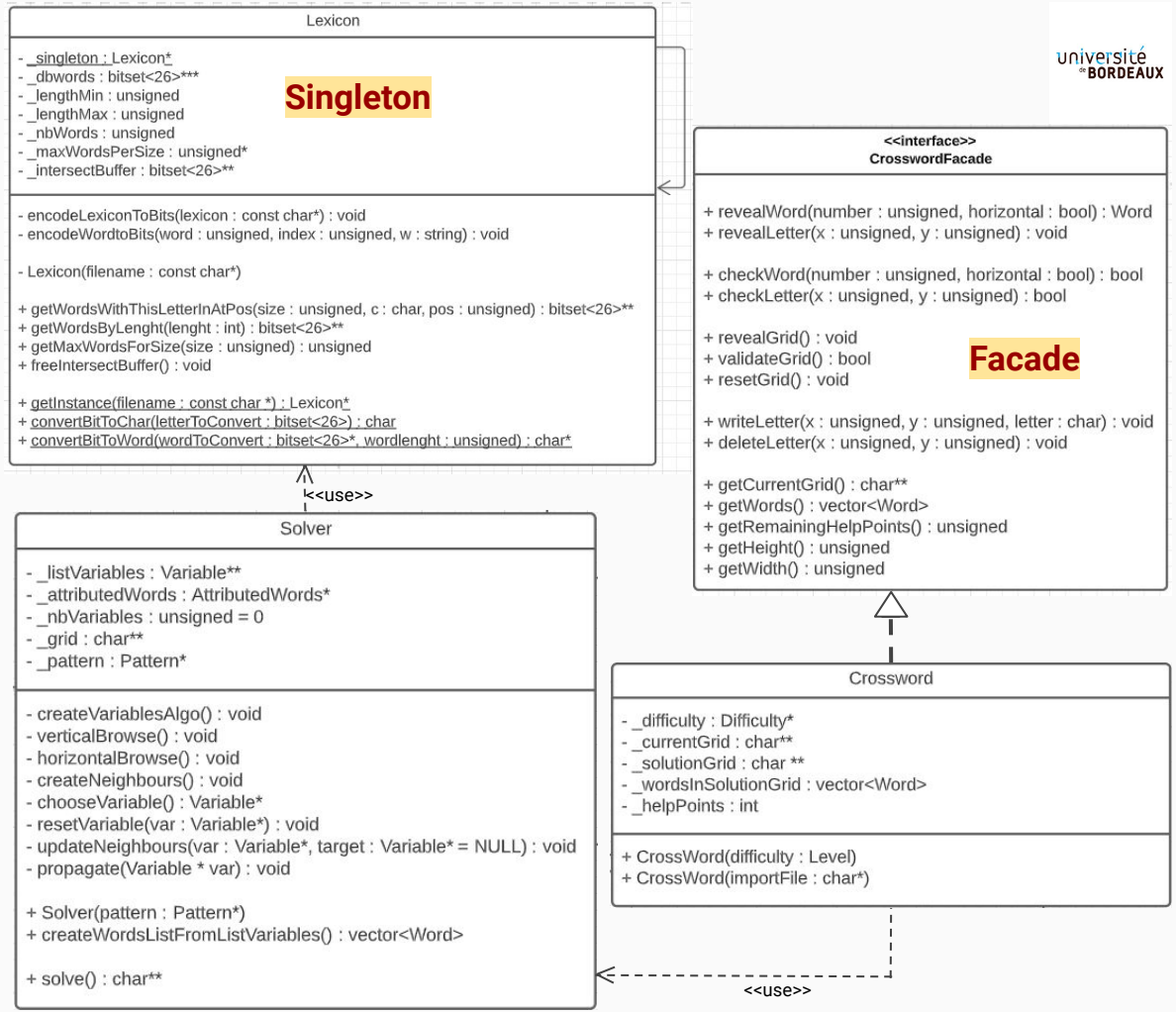


Figure 7 : partie du diagramme de classes

III/ Points techniques et algorithmiques

- Utilisation du langage **C++** (code global)
- **Extraction** et tri de la base de données : Python (bibliothèques Pandas & Numpy)
- Generation via CMake
- Essais non concluants du parallélisme

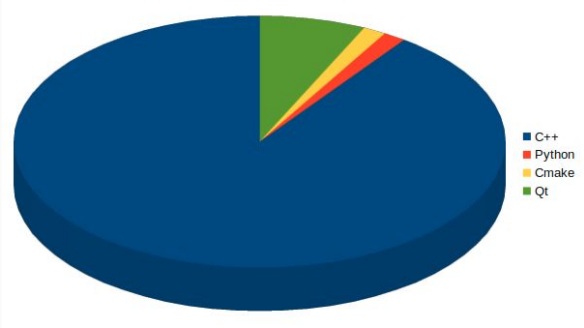


Figure 8 : techniques utilisées au sein du projet

III/ Points techniques et algorithmiques

- Interface Graphique : utilisation de l'API **Qt**

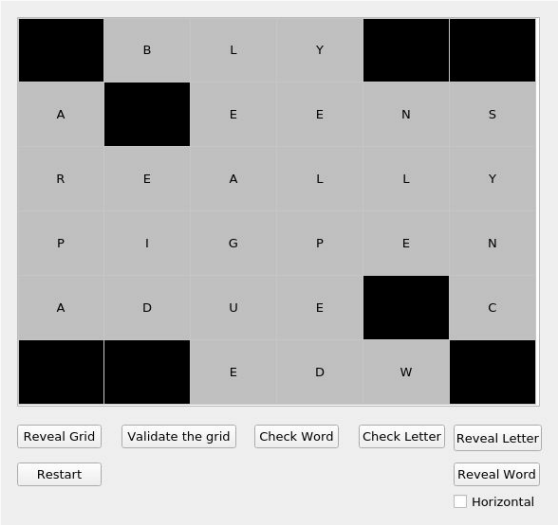


Figure 9 : grille révélée

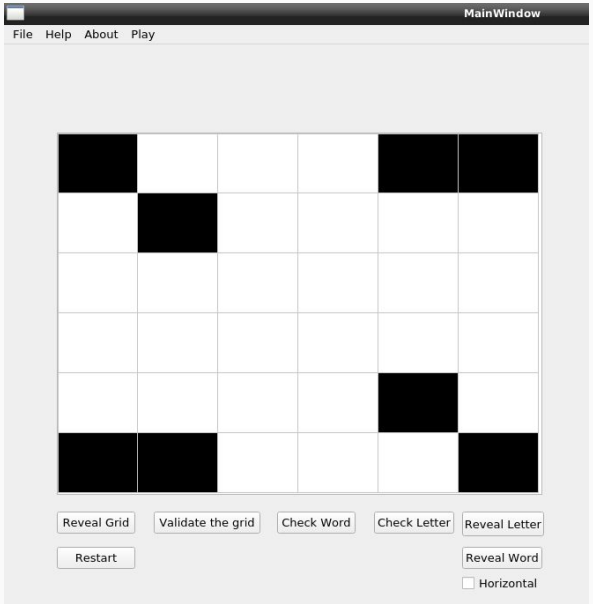


Figure 10 : interface actuelle de l'application

III/ Points techniques et algorithmiques

a) Stockage de la base de données

- Encodage de la base de données en binaire
- Utilisation de la classe **bitset** fournie en C++

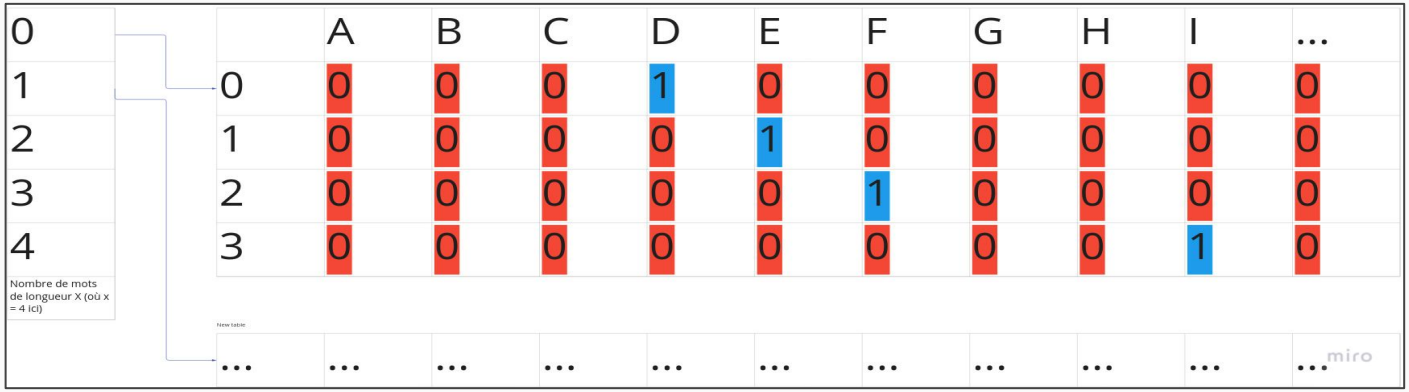
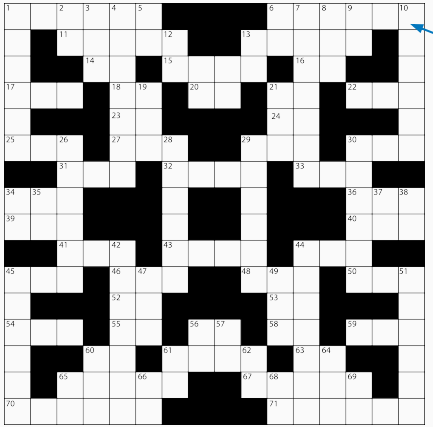


Figure 11 : Exemple de stockage d'un mot au format binaire, mot "défi" ici

III/ Points techniques et algorithmiques

b) Génération de la grille, l'importance des variables



Domaine

Stockage de l'information horizontal / vertical

Champs = une variable

Mot contenu & taille

Position absolue dans la grille

III/ Points techniques et algorithmiques

c) Génération de la grille

Algorithm 2 Algorithme d'attribution des mots aux Variables

```
while Toutes les variables ne sont pas assignées do
    Calculer la taille du domaine de chaque Variable de la grille
    Variable v = variable dont la cardinalité associée au domaine est minimale
    word = premier mot du domaine de v
    if (word != NULL) then
        Valeur de v ← word
        Mettre à jour les domaines des voisins de v
    else
        On retourne en arrière et on teste la valeur suivante pour la variable précédente
    end if
end while
```

IV/ Les premiers résultats

- Temps de **génération** de grille satisfaisants (cf Figure 9)
- Stockage de la base de données efficace en **binaire** (manipulation des jeux de données (très) facilitée par les opérations d'**union** et **intersection** des bitsets

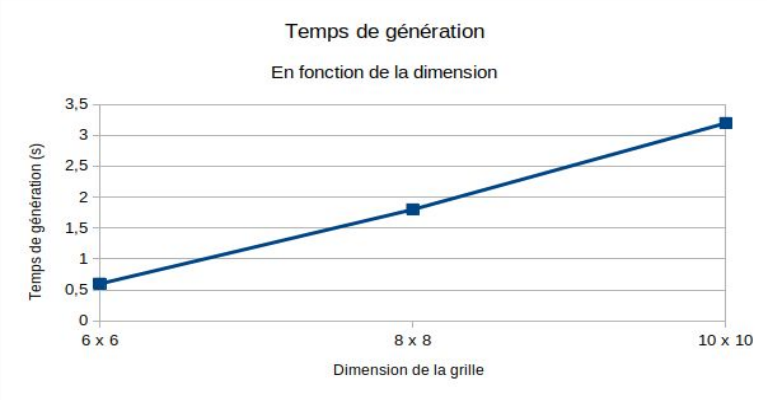


Figure 12 : Les premières performances de temps de génération (dépend des patterns utilisés, ici obtenus avec des patterns ayant beaucoup de cases noires)

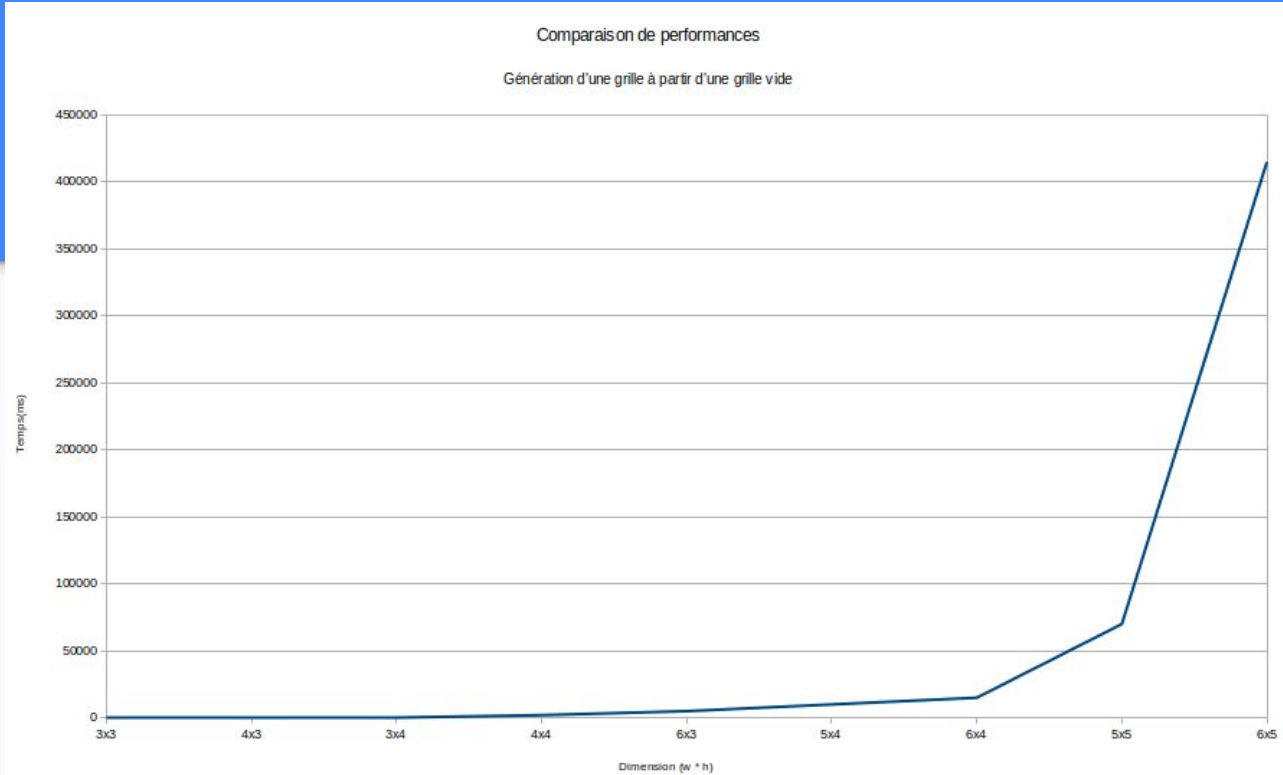
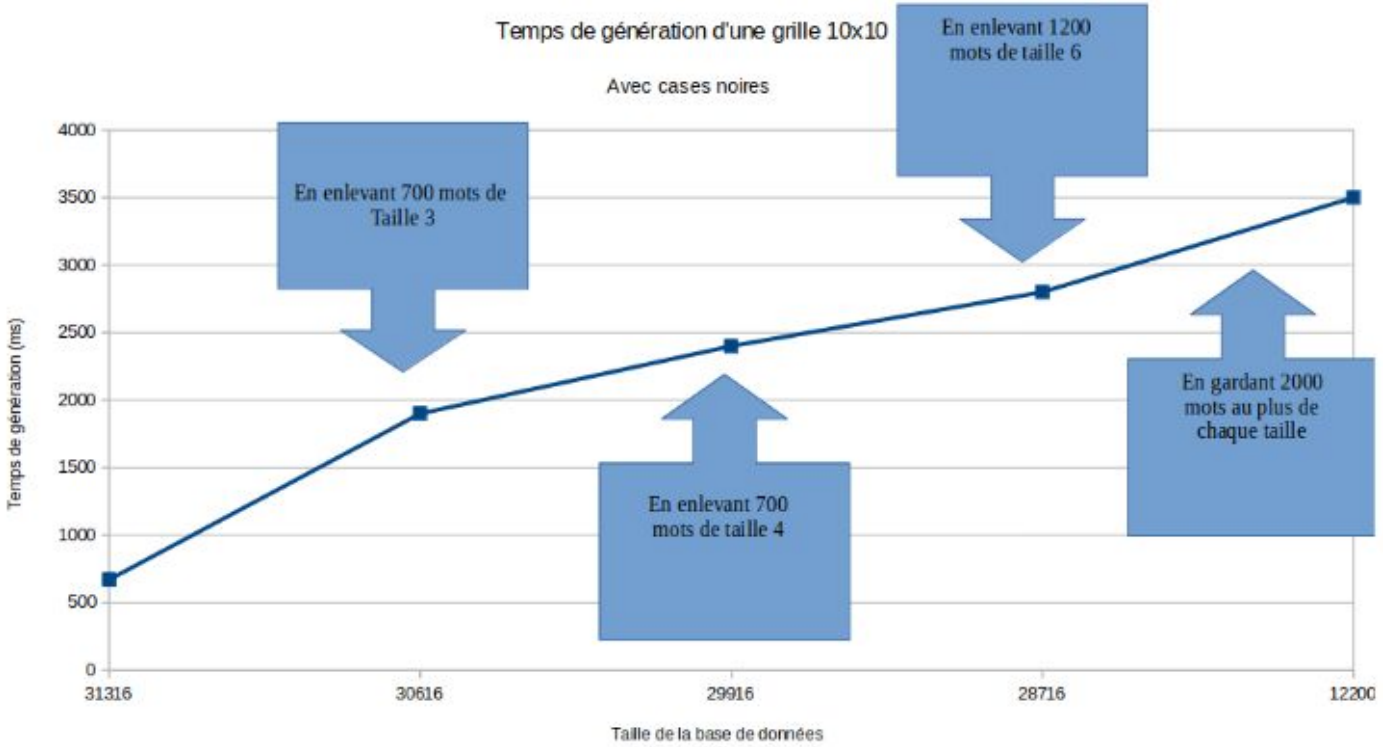


Figure 13 : Les premières performances de temps de génération ***sur des grilles vides***

Nos prochains objectifs

- Affichage des définitions associées aux mots sur l’interface
- Le système de points
- Le refactoring


Temps de génération d'une grille 10x10
Avec cases noires
En enlevant 1200 mots de taille 6
En enlevant 700 mots de Taille 3
En enlevant 700 mots de taille 4
En gardant 2000 mots au plus de chaque taille
Temps de génération (ms)
4000
3500
3000
2500
2000
1500
1000
500
0
31316
30616
29916
28716
12200
Taille de la base de données


user@clone: /mnt/tmp/motscroises/src/build
Fichier Édition Affichage Rechercher Terminal Aide
A # E E N S
R E A L L Y
P I G P E N
A D U E # _
E D W
B L Y #
A # E E N S
R E A L L Y
P I G P E N
A D U E # _
E D W
B L Y #
A # E E N S
R E A L L Y
P I G P E N
A D U E # C
E D W
Grid completed in : 3580.17ms

Patterns

easy1.txt

data > pattern > easy

```
6
6
##___#
_____#
##___#
#___##
#_____
#___##
```

hard2.txt

data > pattern > hard > h

```
10
10
_####____
_____#___
___##____
___#___##
____#____
##_####___
____#____
___#_____
___##____
##____####
```

5_4.txt

data > pattern > tests >

```
5
4
_____
_____
_____
_____
_____
```

word_with_def.txt

data > word_with_def.txt

```
83708  Small plant-part
83709
83710  ROSALIE
83711  Nelson Eddy/Eleanor Powell musical
83712
83713  ROSARIO
83714  "Unstoppable" actress Dawson
83715
83716  ROSEATE
83717  Looking on the bright side
83718
83719  ROSEBUD
83720  Citizen Kane's last word
83721
83722  ROSERED
83723  Snow White's sister
83724
83725  ROSETTA
83726  Stone of note
83727
83728  ROSETTE
83729  Silk ornament
83730
83731  ROSIEST
83732  Least pessimistic
83733
83734  ROSSINI
83735  La Cenerentola composer
83736
83737  ROSSSEA
83738  Arm of the Antarctic
83739
83740  ROSTERS
83741  Lists
83742
83743  ROSTRUM
83744  Speaker's place
83745
83746  ROTATED
83747  Spun
83748
83749  ROTATES
83750  Spins in space
83751
83752  ROTATOR
83753  Kind of muscle
83754
```

word_database.txt

data > word_database.txt

```
2300  YOO
2301  YOS
2302  YOU
2303  YOW
2304  YRS
2305  YSL
2306  YTD
2307  YUK
2308  YUL
2309  YUM
2310  YUP
2311  ZAC
2312  ZAG
2313  ZAK
2314  ZAP
2315  ZAX
2316  ZED
2317  ZEE
2318  ZEN
2319  ZIA
2320  ZIG
2321  ZIN
2322  ZIP
2323  ZIT
2324  ZOA
2325  ZOE
2326  ZOO
2327  ZSA
2328  AAAA
2329  AAAS
2330  AABA
2331  AAHS
2332  AARE
2333  AARP
2334  ABAA
2335  ABAB
2336  ABAD
2337  ABAR
2338  ABAS
```