

LP SIL – Concepteur-Développeur  
en Environnement Distribué  
2017 - 2018

IUT

Robert Schuman

Institut universitaire de technologie

Université de Strasbourg



# PROUST

**DOCUMENTATION  
DU PROJET**

STRATEGIE DE TESTS

GROUPE « OVERKILL LEGACY »

|            |         |
|------------|---------|
| COUTY      | Robin   |
| CROCHEMORE | Thomas  |
| KOEHL      | Dylan   |
| PÉQUIGNOT  | Ysaline |
| WEIBEL     | Lucas   |

# Table des matières

|   |    |
|---|----|
| Table des matières .....                            | 1  |
| Introduction .....                                  | 2  |
| Définition de la stratégie – typologie .....        | 3  |
| Périmètre de test .....                             | 3  |
| Tests fonctionnels .....                            | 3  |
| Fonctions métier .....                              | 3  |
| Prise en compte des accès concurrents .....         | 4  |
| Limites de fonction .....                           | 4  |
| Intégration avec d’autres logiciels .....           | 4  |
| Fonctionnement multi-plateformes .....              | 4  |
| Tests IHM .....                                     | 4  |
| Facilité d’utilisation .....                        | 4  |
| Cohérence et cohésion .....                         | 4  |
| Affichage des erreurs .....                         | 4  |
| Chargements initiaux .....                          | 4  |
| Tests de performance et charge .....                | 5  |
| Tests de sécurité .....                             | 5  |
| Habitations .....                                   | 5  |
| Accès externes .....                                | 5  |
| Fonctionnement en mode dégradé .....                | 5  |
| PCA / PRA .....                                     | 5  |
| Sauvegarde et reprise .....                         | 5  |
| Définition de la stratégie – développements .....   | 6  |
| Analyse statique du code .....                      | 6  |
| Mesures de complexité .....                         | 6  |
| Normes de codage .....                              | 6  |
| Analyse dynamique du code .....                     | 7  |
| Tests “boîte noire” .....                           | 7  |
| Tests “boîte blanche” .....                         | 7  |
| Volume de code couvert .....                        | 8  |
| Nombre d’instructions couvertes .....               | 8  |
| Nombre de branches couvertes .....                  | 8  |
| Nombre de chemins couverts .....                    | 8  |
| Tests de non-régression .....                       | 9  |
| Matrice fréquence/importance .....                  | 9  |
| Conclusion .....                                    | 9  |
| Observations sur le travail effectué .....          | 9  |
| Consolidation des temps composés .....              | 9  |
| Préconisation sur la future campagne de tests ..... | 9  |
| Annexe – Table des illustrations .....              | 10 |

## Introduction

Tout projet nécessite d'être testé, contrôlé, et validé. Dans le monde de l'entreprise, une mise en production ne se résume pas simplement à l'ajout d'une archive sur un serveur : il faut d'abord passer par plusieurs étapes intermédiaires pour s'assurer que ce qui est mis en production est acceptable non seulement d'un point de vue "business", mais aussi d'un point de vue utilisateur.




C'est aussi dans cette optique que nous allons aborder les tests dans le projet ACRObATT, intitulé **PR**évisions météo **Opt**imisées par **U**tilisation de **Si**Tes, ou "**PROUST**" pour faire court. Comme son nom l'indique, ce projet consiste en l'agrégation de données provenant de plusieurs sites météo afin de calculer des prévisions "optimisées", tout en permettant de comparer les différents sites pour déterminer celui qui est le plus fiable. Un aspect d'exploration de données est aussi souhaitable, pour comprendre comment les calculs de fond ont été réalisés.

Dans ce présent dossier, nous allons définir et établir notre stratégie de tests pour ce projet, puis nous nous projetterons dans un avenir où celui-ci serait mis en situation réelle, afin de lister les différents tests à réaliser et leurs durées estimées.

## Définition de la stratégie – typologie

### Périmètre de test

Trois développements ont été réalisés au cours de notre projet ACRObATT :

|  | Ergonomie | Sécurité | Métier | Performances |
|--|-----------|----------|--------|--------------|
| Serveur<br> | NC        |          | 15h    | 6h           |
| Web<br>     | 2h        |          | 6h     | NC           |
| Mobile<br>  | 2h        |          | 6h     |              |

### Tests fonctionnels

#### Fonctions métier

Tableau des fonctionnalités implémentées et testées dans l'application :

|     |  |
|-----|--|
| F1  | Récupération des villes  |
| F2  | Récupération d'une ville par critère (identifiant et latitude/longitude) |
| F3  | Récupération de la ville la plus proche géographiquement (mobile)        |
| F4  | Récupération des pays  |
| F5  | Récupération d'un pays par critère                                       |
| F6  | Récupération des données textuelles                                      |
| F7  | Récupération des délais  |
| F8  | Récupération des propriétés  |
| F9  | Récupération des APIs  |
| F10 | Récupération des distances entre la météo prévue et la météo observée    |
| F11 | Récupération des scores calculés   |
| F12 | Récupération des indices de confiance selon des triplets                 |
| F13 | Récupération des prévisions optimisées à l'heure courante                |
| F14 | Récupération des prévisions par API avec leur indice de confiance        |

## Prise en compte des accès concurrents

Les accès concurrents se font principalement en lecture dans les bases de données. Une “connection pool” ayant été mise en place par le biais d’Hibernate, ceux-ci s’exécutent sans problème à échelle réduite. Cependant, il est possible que les choses doivent changer pour accommoder le système à une plus grande masse d’utilisateurs.

## Limites de fonction

De par la nature du projet, les limites principales concernent la capacité des bases de données elles-mêmes. En-dehors de cela, aucune limite significative n’a été observée.

## Intégration avec d’autres logiciels

Aucune intégration avec des logiciels externe n’est prévue pour ce projet.

## Fonctionnement multi-plateformes

Le fonctionnement multi-plateformes n’est pas prévu de base dans le projet. Cependant, le site web est fonctionnel sous Vivaldi, Mozilla Firefox, Google Chrome et Microsoft Edge. L’application Android a été compilée pour Android 6.1 “Marshmallow”, et devrait fonctionner pour les versions ultérieures jusqu’à Android 8.x “Oreo”.

## Tests IHM

### Facilité d’utilisation

L’ergonomie est primordiale et des efforts ont été faits dans ce sens pour le site web ou l’application mobile. Avec les langages visuels utilisés, connus de tous, l’utilisation des clients est instinctive.

Des tests ont été effectués via l’utilisation des clients par toutes sortes de profils : tous ont compris rapidement à utiliser le site web et l’application et leurs remarques nous ont beaucoup bénéficié, puisque nous n’avons pas leur point de vue “extérieur” au développement.

### Cohérence et cohésion

Comme vu plus haut, les langages visuels modernes sont similaires pour les clients web et mobile, ainsi les deux applicatifs forment un tout cohérent et vont de pair. De plus, la charte graphique utilisée pour les deux clients est la même.

### Affichage des erreurs

Le serveur retourne un élément composé d’un message d’erreur si la requête échoue pour une quelconque raison. Ce message est affiché ou non selon les circonstances sur la page web en question. Un exemple est lorsque le serveur ne trouve aucune donnée correspondant aux critères de l’utilisateur. Dans ce cas, le message « Aucune donnée trouvée » sera affiché sur le client.

### Chargements initiaux

Les chargements initiaux significatifs ont lieu au lancement de l’application mobile, une présentation des fonctionnalités est faite durant ce laps de temps.

## Tests de performance et charge

Aucun test de performance et de charge n'a été prévu pour ce projet.

## Tests de sécurité

### Habilitations

L'accès aux serveurs Tomcat, MySQL, MongoDB, ainsi qu'à la machine virtuelle, sont sécurisés par mot de passe.

### Accès externes

L'accès et la gestion des équipements est gérée par l'Université de Strasbourg et l'IUT Robert Schuman.

### Fonctionnement en mode dégradé

Le fonctionnement en mode dégradé n'est pas prévu pour ce projet.

### PCA / PRA

La machine virtuelle hébergeant le serveur est disponible 24h/24 et 7j/7 sur les serveurs Archipel de l'IUT. Dans le cas d'une défaillance générale, plusieurs actions sont possibles :

- Si le serveur Tomcat est à l'arrêt ou désynchronisé pour une raison quelconque, il est possible de relancer ce service sans affecter le reste de l'application.
- Si la machine elle-même est à l'arrêt, il suffit de la redémarrer puis de s'assurer que les services sont à nouveau en ligne.

Une intégration Slack avec StatusCake a été prévue pour opérer sous la forme d'un ChatOps, notifiant les membres du réseau Slack si le serveur Tomcat ou la machine tombent en panne.

### Sauvegarde et reprise

Compte tenu du matériel fourni et de la configuration actuelle du serveur, aucun mécanisme précis de sauvegarde et reprise n'a été mis en place actuellement. Il est envisageable d'intégrer un système de "point de sauvegarde" des bases de données pour conserver leur contenu dans le cas d'une défaillance.

## Définition de la stratégie – développements

### Analyse statique du code

La dernière analyse du code, réalisée avec SonarQube, donne un aperçu final de l'état statique du projet. Ce rapport indique les choses suivantes :

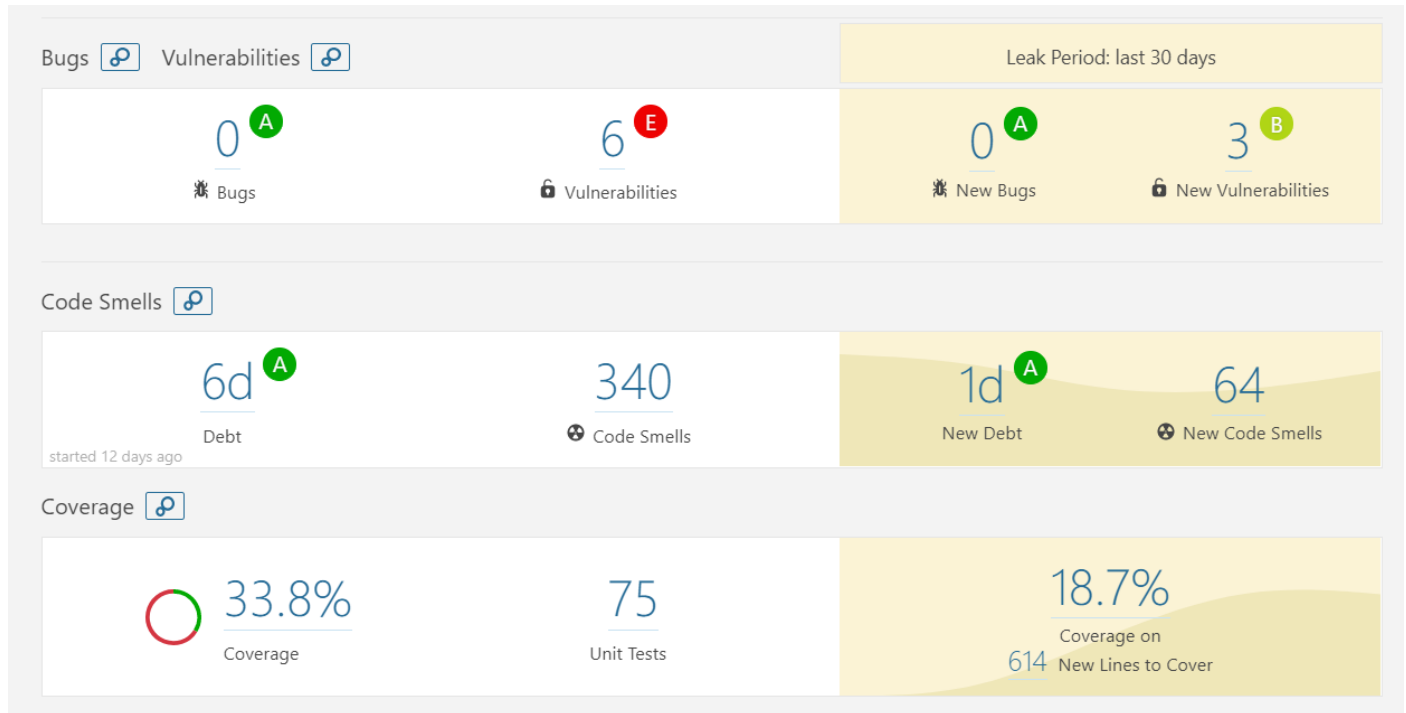


Figure 1 – Informations essentielles du rapport SonarQube

### Mesures de complexité

L'analyse donne une complexité cyclomatique de 992 et une complexité cognitive de 550. Ceci signifie en tant que tel que l'application elle-même est très complexe et relativement difficile à maintenir. Ceci aurait pu être largement amélioré dans un autre contexte que celui dans lequel nous étions.

### Normes de codage

Les principales normes de codage utilisées concernent la présence de commentaires pour justifier et expliquer le code environnant. Autrement, un effort plus poussé a été réalisé côté serveur pour s'assurer que le projet est bien structuré et relativement clair à explorer.

## Analyse dynamique du code

De manière générale, on peut décrire la nature des tests réalisés dans ce projet comme des tests de “boîte grise”, combinant les actions et avantages des deux autres méthodes de test et permettant la création de scénarios de test plus variés et complexes. Cependant, des tests ont néanmoins été réalisés pour chacune des deux méthodes mentionnées.

### Tests “boîte noire”

Lors du développement d’une fonctionnalité majeure, des tests de requête-réponse ont été réalisés pour contrôler le résultat de ces dernières et s’assurer que ce qui est renvoyé correspond à ce qui est attendu d’un point de vue utilisateur. Pour appuyer cette procédure, des “loggers”, outils servant à créer un historique des événements, ont été créés, et indiquent les opérations et emplacements précis de chaque étape majeure de traitement au sein du code serveur.

Un exemple de logging se trouve ci-dessous :

```
[DEBUG] 2018-06-03 09:54:01.046 [ComparativeDataDAO.getRawIndexData:312] - Chosen criteria : {datetime=null, is_forecast=false, delay=24, city=Strasbourg, property=temperature, api=null, datetime_to=null}

[DEBUG] 2018-06-03 09:54:01.061 [ComparativeDataDAO.getRawScoreData:236] - Chosen criteria : {datetime=Mon Feb 12 01:00:00 CET 2018, is_forecast=false, delay=24, city=Strasbourg, property=temperature, api=null, datetime_to=Sun Jun 03 09:00:00 CEST 2018}

[DEBUG] 2018-06-03 09:54:01.065 [ComparativeDataDAO.getRawScoreData:301] - select distinct generatedAlias0.api, coalesce((1/avg(generatedAlias0.distance)), 0.01D) from ComparativeData as generatedAlias0 where ( generatedAlias0.location=:param0 ) and ( generatedAlias0.prop=:param1 ) and ( generatedAlias0.delay=24 ) and ( generatedAlias0.date_req between :param2 and :param3 ) group by generatedAlias0.api

[INFO ] 2018-06-03 09:54:01.079 [ComparativeDataDAO.getRawIndexData:329] - Calculated score sum = 1.0394465676743927

[DEBUG] 2018-06-03 09:54:01.081 [ComparativeDataDAO.getRawIndexData:392] - select distinct generatedAlias0.api, (coalesce((1/avg(generatedAlias0.distance)), 0.01D)/1.0394465676743927D) from ComparativeData as generatedAlias0 where ( ( generatedAlias0.date_req<:param0 ) and ( generatedAlias0.location=:param1 ) ) and ( generatedAlias0.prop=:param2 ) ) and ( generatedAlias0.delay=24 ) group by generatedAlias0.api order by (coalesce((1/avg(generatedAlias0.distance)), 0.01D)/1.0394465676743927D) desc

[DEBUG] 2018-06-03 09:54:01.092 [ComparativeDataModule.convertData:40] - rowHeaders = [APIXU, OPENWEATHERMAP, WUNDERGROUND, DARKSKY]

[DEBUG] 2018-06-03 09:54:01.093 [ComparativeDataModule.convertData:41] - colHeaders = [API, Indice de confiance]
```

### Tests “boîte blanche”

Afin de vérifier le fonctionnement de l’application côté serveur, plusieurs classes de test ont été créées avec JUnit 5. Ces classes se trouvent dans le dossier “test” au sein de l’architecture serveur.

Les tests concernent principalement les DAOs pour des raisons simples : l’accès à la base de données étant critique dans toute application, et donc par déduction dans celle-ci, il faut pouvoir s’assurer que la récupération des informations se fait exactement comme demandé.

Une autre catégorie de tests concerne les unités de transformation de données de MongoDB vers MySQL, appelés “parsers”. Cette fonction critique doit impérativement être testée pour assurer son opération normale.



## Volume de code couvert

Le rapport complet de la couverture de code, basé uniquement sur les tests unitaires implémentés avec JUnit sur une base MySQL de test préalablement vidée, se trouve ci-dessous :

### proust



















































| Element                                       | Missed Instructions   | Cov. | Missed Branches  | Cov.  | Missed | Cxty  | Missed | Lines | Missed | Methods | Missed | Classes |
|---|---|------|--|-------|--------|-------|--------|-------|--------|---------|--------|---------|
| org.acrobatt.project.dao.mysql.analysis       |    | 24 % |   | 16 %  | 87     | 99    | 274    | 364   | 20     | 28      | 0      | 1       |
| org.acrobatt.project.dao.mysql                |    | 46 % |   | 23 %  | 77     | 181   | 360    | 843   | 42     | 135     | 1      | 12      |
| org.acrobatt.project.controllers              |    | 78 % |   | 43 %  | 69     | 93    | 261    | 468   | 41     | 45      | 6      | 7       |
| org.acrobatt.project.services                 |    | 14 % |   | 7 %   | 100    | 110   | 201    | 240   | 44     | 53      | 6      | 8       |
| org.acrobatt.project.services.rawdata.modules |    | 0 %  |   | 0 %   | 66     | 66    | 199    | 199   | 27     | 27      | 4      | 4       |
| org.acrobatt.project.apiparsers.parsers       |    | 55 % |   | 83 %  | 4      | 32    | 158    | 376   | 0      | 20      | 0      | 4       |
| org.acrobatt.project.dao.mysql.cache          |    | 11 % |   | 19 %  | 38     | 49    | 182    | 204   | 22     | 31      | 0      | 3       |
| org.acrobatt.project.services.rawdata         |    | 0 %  |   | 0 %   | 39     | 39    | 91     | 91    | 17     | 17      | 5      | 5       |
| org.acrobatt.project.weatherdata              |    | 10 % |   | 6 %   | 25     | 29    | 73     | 88    | 10     | 13      | 2      | 3       |
| org.acrobatt.project.weatherapi               |    | 7 %  |   | 14 %  | 13     | 17    | 57     | 65    | 7      | 10      | 1      | 2       |
| org.acrobatt.project.model.mysql              |    | 57 % |   | 33 %  | 66     | 138   | 98     | 248   | 60     | 129     | 0      | 12      |
| org.acrobatt.project.model.mysql.cache        |    | 0 %  |   | 0 %   | 46     | 46    | 100    | 100   | 44     | 44      | 3      | 3       |
| org.acrobatt.project.utils                    |    | 41 % |   | 25 %  | 46     | 60    | 60     | 105   | 20     | 32      | 4      | 8       |
| org.acrobatt.project.services.auth            |    | 0 %  |   | 0 %   | 18     | 18    | 43     | 43    | 7      | 7       | 1      | 1       |
| org.acrobatt.project.utils.auth               |    | 0 %  |   | 0 %   | 14     | 14    | 37     | 37    | 12     | 12      | 2      | 2       |
| org.acrobatt.project.utils.enums              |    | 0 %  | n/a  | n/a   | 12     | 12    | 15     | 15    | 12     | 12      | 3      | 3       |
| org.acrobatt.project.utils.struct             |    | 0 %  |   | 0 %   | 22     | 22    | 26     | 26    | 12     | 12      | 2      | 2       |
| org.acrobatt.project.dao.mongo                |    | 9 %  |   | 25 %  | 6      | 8     | 33     | 38    | 4      | 6       | 0      | 1       |
| org.acrobatt.project.filters                  |    | 0 %  |   | 0 %   | 7      | 7     | 35     | 35    | 6      | 6       | 2      | 2       |
| org.acrobatt.project.mongo                    |    | 48 % |   | 33 %  | 24     | 33    | 19     | 46    | 3      | 12      | 0      | 2       |
| org.acrobatt.project.apiparsers               |    | 33 % |   | 21 %  | 12     | 18    | 16     | 30    | 6      | 11      | 1      | 3       |
| org.acrobatt.project.model.mysql.analysis     |    | 8 %  | n/a  | n/a   | 17     | 19    | 21     | 23    | 17     | 19      | 0      | 2       |
| org.acrobatt.project.model.mongo              |    | 49 % |   | 0 %   | 10     | 22    | 7      | 31    | 9      | 21      | 1      | 2       |
| org.acrobatt.project.dto                      |    | 0 %  | n/a  | n/a   | 9      | 9     | 9      | 9     | 9      | 9       | 5      | 5       |
| org.acrobatt.project.utils.db                 |   | 57 % |  | 100 % | 2      | 6     | 10     | 19    | 2      | 5       | 0      | 1       |
| org.acrobatt.project.auth                     |  | 0 %  | n/a  | n/a   | 6      | 6     | 7      | 7     | 6      | 6       | 1      | 1       |
| org.acrobatt.project.listeners                |  | 0 %  | n/a  | n/a   | 3      | 3     | 8      | 8     | 3      | 3       | 1      | 1       |
| org.acrobatt.project.except                   |  | 0 %  | n/a  | n/a   | 2      | 2     | 4      | 4     | 2      | 2       | 1      | 1       |
| Total   | 12 358 of 21 579  | 42 % | 685 of 852   | 19 %  | 840    | 1 158 | 2 404  | 3 762 | 464    | 727     | 52     | 101     |

Figure 2 - Rapport de couverture de code JaCoCo

### Nombre d'instructions couvertes

Basé sur le rapport ci-dessus: sur 21 579 instructions, 12 358 ont été oubliées, donc 9221 ont été couvertes.

### Nombre de branches couvertes

Basé sur le rapport ci-dessus: sur 852 branches, 685 ont été oubliées, donc 167 ont été couvertes.

### Nombre de chemins couverts

Le nombre de chemins couverts est indisponible pour l'outil de couverture de tests sélectionné.

## Tests de non-régression

Cette catégorie de tests n'a pas été prévue pour ce projet.

### Matrice fréquence/importance

|         | Stratégique | Important | Secondaire |
|---------|-------------|-----------|------------|
| Forte   | 1           | 2         | 3          |
| Moyenne | 1           | 3         | 4          |
| Faible  | 2           | 4         | 4          |

|         | Stratégique             | Important | Secondaire |
|---------|-------------------------|-----------|------------|
| Forte   | F6,F7,F8,F9,F12,F13,F14 | F1,F2     |            |
| Moyenne |                         |           |            |
| Faible  | F3, F10,F11             | F4,F5     |            |

### Conclusion

#### Observations sur le travail effectué

La partie test automatisé a été essentiellement faite sur le serveur Tomcat, nous prenons quelques minutes à vérifier ces tests. Certains tests côté serveur, et les tests web et mobile sont fait manuellement, nous mettons donc plus de temps à effectuer ces tests. Pour une application parfaite, nous devons automatiser le tout, car l'erreur est humaine. Malheureusement, comme nous ne pouvons pas tout automatiser en si peu de temps, notre stratégie s'est faite sur le point fonctionnel de notre serveur.

#### Consolidation des temps composés

Dylan et Robin, ont fait des tests unitaires et d'intégration de notre serveur à l'aide de JUnit. Dylan a utilisé SonarQube pour corriger les vulnérabilités du code, Robin a utilisé StatusCake pour prévenir d'une panne de serveur.

Pour le web, Ysaline et Lucas ont mis en place des pages contenant chacun de leurs composants. Cela leur permet de tester manuellement chacun de leurs composants, avant de tester toute l'application.

Pour le mobile, Thomas teste manuellement directement sur son application.

#### Préconisation sur la future campagne de tests

Il faudrait automatiser certains tests avec un jeu de données prédéfinis dans le serveur, pointant sur une base de données spécialisée dans les tests. Pour le web et le mobile, nous pouvons aussi utiliser Karma pour automatiser non l'affichage, mais les données exploitées et affichées dans leur interface. L'affichage est difficilement testable car une application en évolution change beaucoup d'affichage, alors que les données non.

## Annexe – Table des illustrations

|  |   |
|--|---|
| Figure 1 – Informations essentielles du rapport SonarQube..... | 6 |
| Figure 2 - Rapport de couverture de code JaCoCo .....          | 8 |

## Annexe – Index

|                       |         |                  |                  |
|-----------------------|---------|------------------|------------------|
| ChatOps .....         | 5       | MongoDB .....    | 5, 7             |
| complexité .....      | 1, 6    | MySQL.....       | 5, 7, 8          |
| couverture .....      | 8, 10   | SonarQube .....  | 6, 9, 10         |
| fonctionnalités ..... | 3, 4    | StatusCake ..... | 5, 9             |
| Hibernate .....       | 4       | tests .....      | 1, 2, 4, 7, 8, 9 |
| JUnit .....           | 7, 8, 9 | Tomcat .....     | 5, 9             |