# Rapport test unitaires

Nous avons décidé de faire les tests unitaires sur l'API de notre projet. En effet, c'est sur ce projet que se concentre l'essentiel de l'intelligence du projet (création, modification et suppression des éléments, vérification des données, etc).

L'API est créé en TypeScript et utilise les librairies Express et Sequelize. Nous avons donc choisi `ts-jest` pour nos tests unitaires, couplé avec le package `@jest-mock/express` afin de faciliter le test de nos controllers express et `faker` afin de créer un jeu de données réaliste.

## Qu'est ce qui a été testé ?

Nous avons fait le choix de tester les controlleurs, car il représente la quasi totalité de l'intelligence de l'api. De plus, tester les controlleurs permet de tester indirectement les modèles de la base de données.

Plus spécifiquement, nous avons décidé de tester les controlleurs gérant les utilisateurs et les voyages.

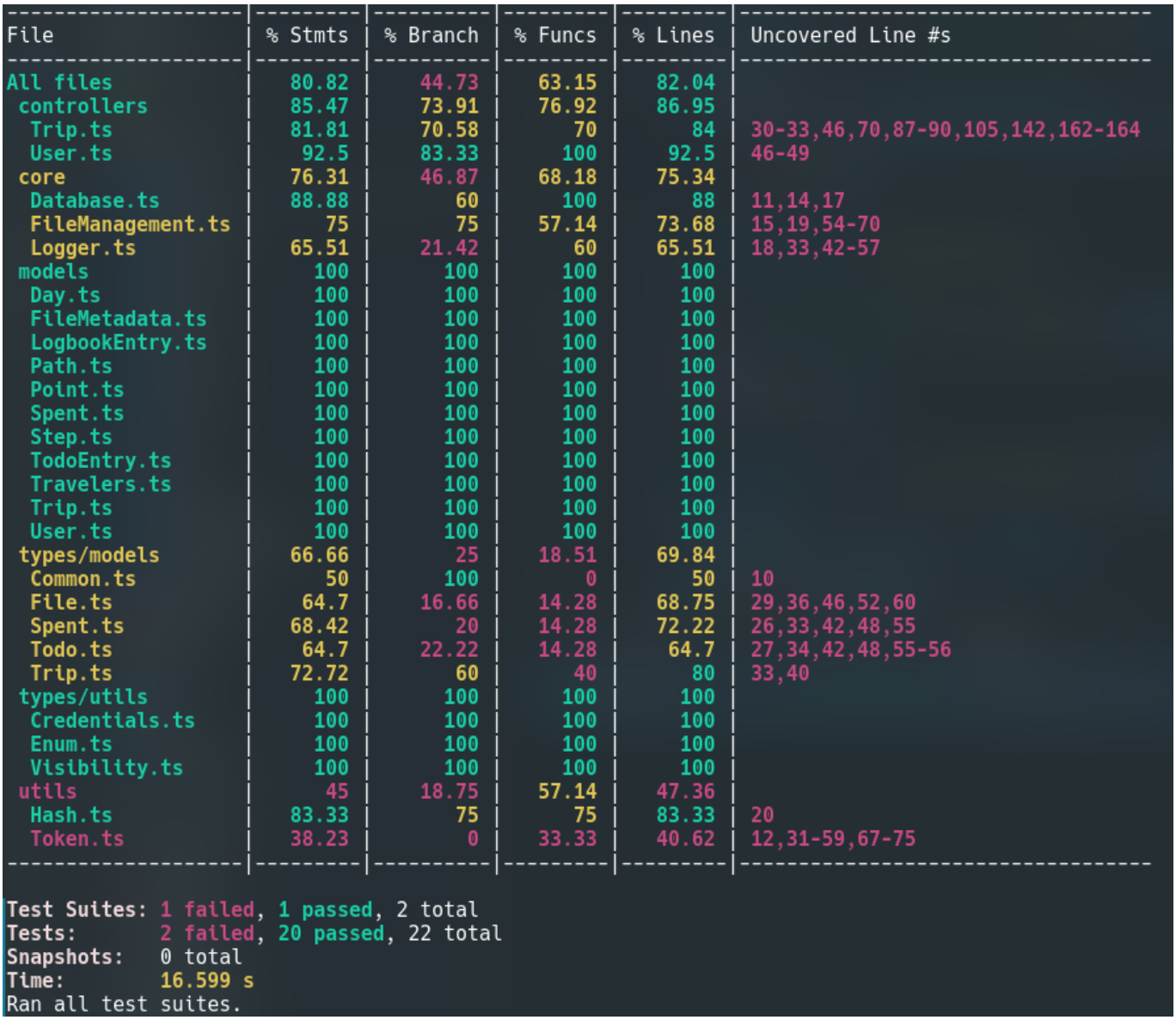
## Quelles leçons en tirer ?

Mettre en place ces tests unitaires nous a permis de mettre en valeur certains défauts de conception dans le code de ces controllers.
Par exemple, nous avons pu remarquer qu'aucune vérification n'était effectuée sur la forme des informations rentrées en base (par exemple, vérifier qu'une adresse mail à un format valide).
Nous avons également remarqué que certains codes d'erreurs renvoyés n'était pas les bons.

## Quels bénéfices en tirer ?

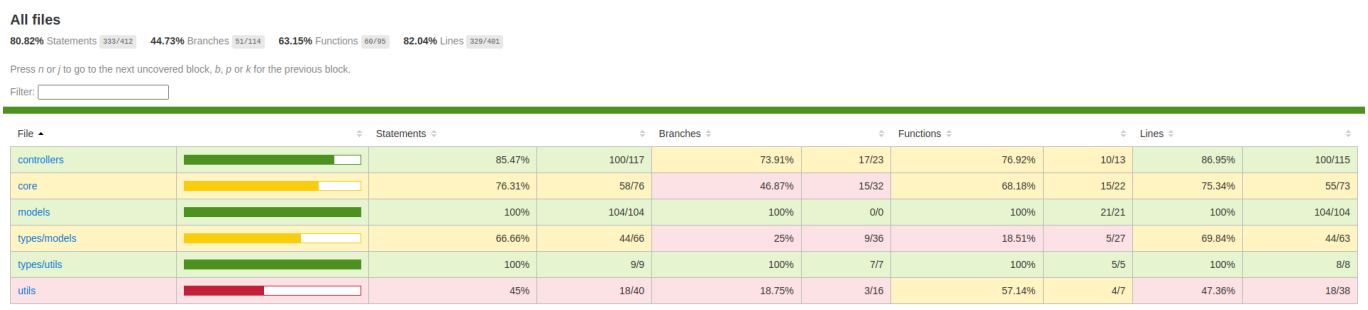
L'avantage de `jest` est qu'il est très facile à intégrer dans une CI. Nous avons donc ajouté une Github Actions pour automatiquement lancer les tests unitaires depuis une CI. Désormais, des tests sont mis en place lorsqu'on ouvre une pull request vers notre branche `master`. Cela évite de mettre en production du code non fonctionnel qui casserait une fonctionnalité.

## Rapport de couverture du code par les tests



| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
|---|---|---|---|---|---|
| All files | 80.82 | 44.73 | 63.15 | 82.04 | |
| controllers | 85.47 | 73.91 | 76.92 | 86.95 | |
| Trip.ts | 81.81 | 70.58 | 70 | 84 | 30-33,46,70,87-90,105,142,162-164 |
| User.ts | 92.5 | 83.33 | 100 | 92.5 | 46-49 |
| core | 76.31 | 46.87 | 68.18 | 75.34 | |
| Database.ts | 88.88 | 60 | 100 | 88 | 11,14,17 |
| FileManagement.ts | 75 | 75 | 57.14 | 73.68 | 15,19,54-70 |
| Logger.ts | 65.51 | 21.42 | 60 | 65.51 | 18,33,42-57 |
| models | 100 | 100 | 100 | 100 | |
| Day.ts | 100 | 100 | 100 | 100 | |
| FileMetadata.ts | 100 | 100 | 100 | 100 | |
| LogbookEntry.ts | 100 | 100 | 100 | 100 | |
| Path.ts | 100 | 100 | 100 | 100 | |
| Point.ts | 100 | 100 | 100 | 100 | |
| Spent.ts | 100 | 100 | 100 | 100 | |
| Step.ts | 100 | 100 | 100 | 100 | |
| TodoEntry.ts | 100 | 100 | 100 | 100 | |
| Travelers.ts | 100 | 100 | 100 | 100 | |
| Trip.ts | 100 | 100 | 100 | 100 | |
| User.ts | 100 | 100 | 100 | 100 | |
| types/models | 66.66 | 25 | 18.51 | 69.84 | |
| Common.ts | 50 | 100 | 0 | 50 | 10 |
| File.ts | 64.7 | 16.66 | 14.28 | 68.75 | 29,36,46,52,60 |
| Spent.ts | 68.42 | 20 | 14.28 | 72.22 | 26,33,42,48,55 |
| Todo.ts | 64.7 | 22.22 | 14.28 | 64.7 | 27,34,42,48,55-56 |
| Trip.ts | 72.72 | 60 | 40 | 80 | 33,40 |
| types/utils | 100 | 100 | 100 | 100 | |
| Credentials.ts | 100 | 100 | 100 | 100 | |
| Enum.ts | 100 | 100 | 100 | 100 | |
| Visibility.ts | 100 | 100 | 100 | 100 | |
| utils | 45 | 18.75 | 57.14 | 47.36 | |
| Hash.ts | 83.33 | 75 | 75 | 83.33 | 20 |
| Token.ts | 38.23 | 0 | 33.33 | 40.62 | 12,31-59,67-75 |

```
Test Suites: 1 failed, 1 passed, 2 total
Tests:       2 failed, 20 passed, 22 total
Snapshots:   0 total
Time:        16.599 s
Ran all test suites.
```



**All files**

**80.82%** Statements 333/412 **44.73%** Branches 51/114 **63.15%** Functions 60/95 **82.04%** Lines 329/401

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

| File | | Statements | | Branches | | Functions | | Lines | |
|---|---|---|---|---|---|---|---|---|---|
| controllers | | 85.47% | 100/117 | 73.91% | 17/23 | 76.92% | 10/13 | 86.95% | 100/115 |
| core | | 76.31% | 58/76 | 46.87% | 15/32 | 68.18% | 15/22 | 75.34% | 55/73 |
| models | | 100% | 104/104 | 100% | 0/0 | 100% | 21/21 | 100% | 104/104 |
| types/models | | 66.66% | 44/66 | 25% | 9/36 | 18.51% | 5/27 | 69.84% | 44/63 |
| types/utils | | 100% | 9/9 | 100% | 7/7 | 100% | 5/5 | 100% | 8/8 |
| utils | | 45% | 18/40 | 18.75% | 3/16 | 57.14% | 4/7 | 47.36% | 18/38 |

# Code source

## Code source de l'api

Les tests sont disponible dans le dossier `__tests__`.