Conception des grilles

# **Introduction**

Dans ce rapport, nous abordons la conception d'un aspect clé de notre application de gestion de sessions de conduite : le système de grille de notation. La flexibilité et la robustesse de ce système sont primordiales pour évaluer efficacement les performances des élèves et enrichir leur expérience d'apprentissage. Il doit non seulement offrir un choix de modèles de grille, mais aussi permettre la création de nouvelles grilles et la transmission des données vers des Learning Record Store (LRS).

Nous avons exploré diverses approches pour développer ce système, en pesant leurs avantages et inconvénients en termes de flexibilité, complexité, performances, temps de développement et facilité d'utilisation. Ce rapport présente les résultats de cette exploration, y compris les recommandations pour la mise en œuvre du système de grille de notation.

L'objectif final est de créer un système qui facilite le travail des instructeurs, optimise l'apprentissage des élèves et augmente la valeur globale de notre application dans le domaine de la formation à la conduite. [Consultez d’abord ces quelques conseils pour vous aider à mettre en forme votre document. Vous verrez, c’est très facile.]

# 

# **Recherche**

Dans le but de concevoir un système de grille de notation efficace et adaptable, nous avons entrepris une recherche extensive pour examiner les différentes solutions possibles. Notre objectif était de trouver une solution qui répondrait à nos besoins spécifiques tout en restant flexible pour les utilisateurs finaux et efficace en termes de performance et de maintenabilité.

1. Modification de la structure de la base de données

La première solution que nous avons considérée implique une refonte de la structure de notre base de données pour accommoder le système de grille de notation. Cela signifierait la création de nouvelles tables pour représenter les grilles, les lignes de la grille, et les cellules individuelles [1].

Avantages : Cette approche offre une flexibilité maximale en ce qui concerne la configuration des grilles. Elle nous permet de modéliser précisément le système de grille à l'intérieur de la base de données elle-même, ce qui pourrait conduire à une meilleure performance et à une plus grande facilité de maintenance.



[1] [Laravel Migrations](https://laravel.com/docs/8.x/migrations" \t "_new)

2. Déplacement de la logique vers le côté client

Une autre solution que nous avons explorée consiste à déplacer une partie significative de la logique de la grille vers le côté client [1].

Avantages : Cette approche pourrait rendre l'application plus interactive, en permettant aux utilisateurs de manipuler directement les grilles de notation. Elle peut également réduire la charge sur le serveur, puisque une grande partie du traitement serait effectuée côté client [2].

Inconvénients : Toutefois, cette solution pourrait également rendre l'application plus complexe et difficile à maintenir. De plus, elle dépend beaucoup de la performance du navigateur de l'utilisateur et pourrait conduire à des expériences utilisateur incohérentes.

Globalement, bien que cette solution puisse offrir une meilleure interactivité, elle pourrait également augmenter la complexité et la charge de maintenance de l'application.

[1] [Client-side vs Server-side Rendering in JavaScript Applications](https://www.toptal.com/javascript/client-side-vs-server-side-rendering" \t "_new)

[2] [How to build a Client-Side App with JavaScript](https://www.freecodecamp.org/news/how-to-build-a-client-side-app-with-javascript-f3208e5146c4/" \t "_new)

3. Utilisation d'une bibliothèque externe

La dernière solution que nous avons envisagée est l'utilisation d'une bibliothèque externe pour gérer les grilles [1].

Avantages : L'utilisation d'une bibliothèque externe bien conçue et bien supportée pourrait nous faire gagner beaucoup de temps de développement. De nombreuses bibliothèques proposent des fonctionnalités sophistiquées pour la gestion des grilles, ce qui pourrait augmenter la flexibilité et la robustesse de notre système de grille [2].

Inconvénients : Cependant, cette solution dépend fortement de la bibliothèque externe choisie. Si la bibliothèque cesse d'être maintenue ou présente des problèmes de compatibilité, cela pourrait poser des problèmes pour notre application.

En résumé, l'utilisation d'une bibliothèque externe pourrait être une option viable si nous trouvons une bibliothèque qui répond à nos besoins spécifiques et qui est activement maintenue.

[1] [Top JavaScript Grid Libraries for 2023](https://blog.bitsrc.io/top-javascript-grid-libraries-for-2021-40faa2197fb8" \t "_new)

[2] [How to Choose a JavaScript Library for Your Web Project](https://raygun.com/blog/how-to-choose-javascript-library/" \t "_new)

Décision finale

Après avoir soigneusement pesé les avantages et les inconvénients de chaque solution, nous avons décidé de choisir la première solution : Modification de la structure de la base de données. Une des principales raisons de ce choix est que notre application n'est pas encore en production, ce qui nous offre une flexibilité considérable pour modifier la structure de la base de données. De plus, cette approche nous donne une grande flexibilité pour concevoir le système de grille selon nos besoins spécifiques, et elle est susceptible d'offrir de bonnes performances et une facilité de maintenance.

Dans la section suivante, nous discuterons des étapes que nous prévoyons de prendre pour mettre en œuvre cette solution et comment nous prévoyons de surmonter les défis potentiels que nous pourrions rencontrer.

**Mise en œuvre de la solution**

Suite à notre recherche et à notre analyse, nous avons choisi d'opter pour la solution impliquant une modification de la structure de la base de données pour implanter le système de grilles. Notre mise en œuvre se déroulera comme suit :

**Conception de la nouvelle structure de la base de données**

Notre première étape consistera à concevoir la nouvelle structure de notre base de données. Cela comprendra la création de nouvelles tables pour les modèles de grilles, les grilles, et les cellules de grille, et également la modification de certaines des tables existantes pour accueillir ces changements. Nous nous servirons d'outils de modélisation de données pour visualiser cette nouvelle structure et définir les relations entre les différentes tables.

Un schéma de cette nouvelle structure de base de données est disponible en annexe.

(Modelisation-DataBase-V5. Annexe 1)

**Création des migrations de la base de données**

Après avoir établi un modèle clair de notre nouvelle structure de données, nous créerons des migrations pour ces nouvelles tables avec Laravel. Cela nous permettra de garder un historique des modifications apportées à notre base de données et de faciliter les futures mises à jour.

**Mise à jour des modèles**

Une fois que les migrations seront créées, nous mettrons à jour les modèles Laravel correspondants pour refléter la nouvelle structure de la base de données. Cela inclura la création de nouveaux modèles pour les tables des grilles et la modification des modèles existants si nécessaire.

**Test des modifications**

Après avoir effectué ces modifications, nous réaliserons une série de tests pour nous assurer que la nouvelle structure de la base de données fonctionne comme prévu. Cela comprendra des tests unitaires pour les nouvelles fonctionnalités et des tests d'intégration pour vérifier que l'ensemble du système fonctionne correctement.

**Création des grilles**

Nous définirons les critères et les thèmes pour chacune des trois grilles. Nous utiliserons des outils de conception pour visualiser comment ces grilles seront présentées à l'utilisateur.

Une maquette pour la grille est également disponible en annexe.

(Maquette-grille-acp. Annexe 2)

**Développement des points d'accès API**

Nous développerons des points d'accès API pour gérer ces grilles. Ces API permettront aux utilisateurs de récupérer, de créer, de modifier et de supprimer des grilles.

Un diagramme de séquence pour la gestion d'une journée de formation/session en utilisant ces points d'accès API est également disponible en annexe.

(Diagramme-sequence-session. Annexe 3)

**Intégration des API avec l'interface utilisateur**

Une fois que nos API seront opérationnelles, nous les intégrerons avec l'interface utilisateur de notre application. Cela signifie que nous devrons utiliser JavaScript (ou un framework JavaScript) pour effectuer des appels API depuis le navigateur et afficher les résultats à l'utilisateur.

**Enregistrement des données de la grille**

Lorsqu'un utilisateur soumet une grille, nous utiliserons nos API pour enregistrer ces données dans la base de données. Cela impliquera de transformer les données de la grille en un format qui peut être stocké dans la base de données, puis de faire un appel API pour créer ou mettre à jour l'enregistrement correspondant.

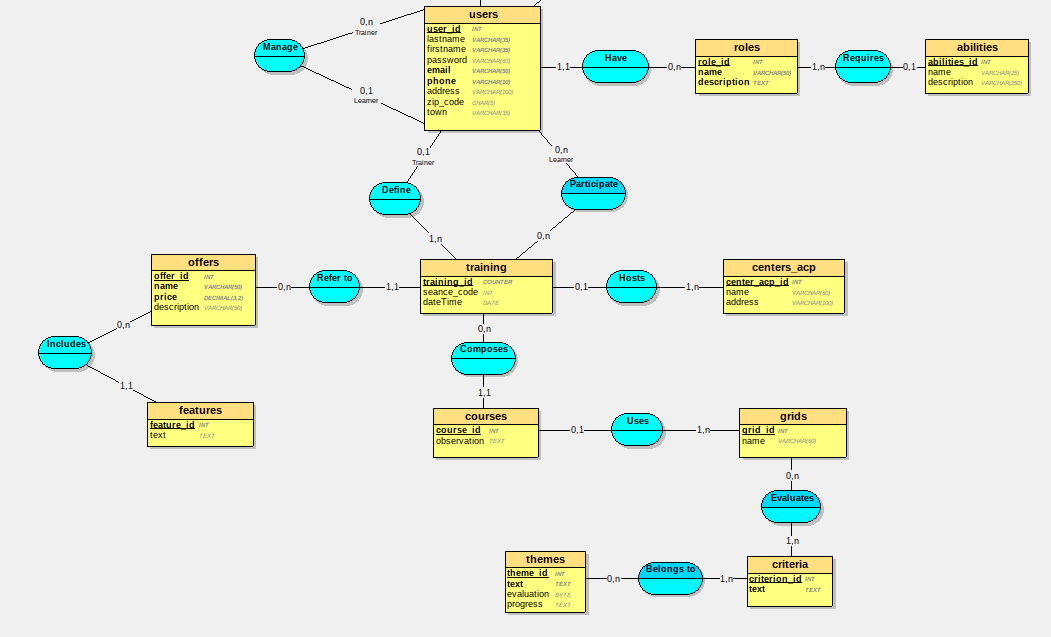
**Conclusion**

Ce rapport a examiné le défi de mettre en œuvre un système de grilles dans notre application Laravel. Après avoir exploré plusieurs options, nous avons choisi de modifier la structure de la base de données, une décision dictée par sa flexibilité et sa capacité d'intégration.

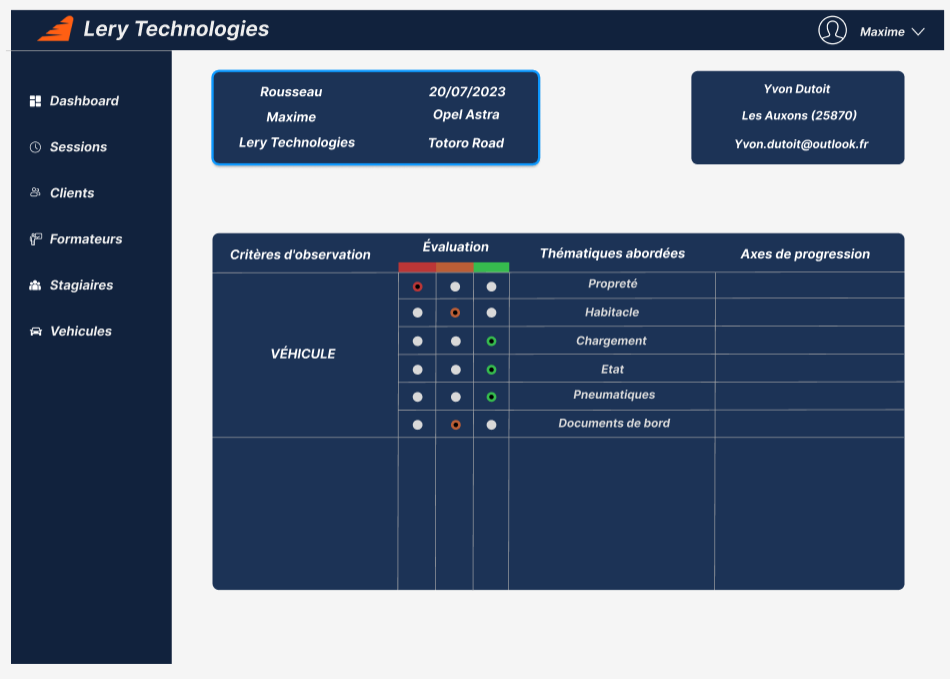
La mise en œuvre de cette solution comprendra la modification de la base de données, le développement de points d'accès API, et l'intégration avec l'interface utilisateur. Bien que cette approche présente certains défis, comme la gestion de plusieurs modèles de grilles, nous sommes confiants que notre planification rigoureuse et notre exécution méticuleuse nous permettront de les surmonter.

**Annexes**

**Annexe 1 Modelisation-DataBase-V5**



**Annexe 2 Maquette-grille-acp**



**Annexe 3 Diagramme-sequence-session**

