Tchat Hodinfosse : cheatsheet

Table des matières

[Introduction 3](#_Toc171607408)

[Google Authentication 4](#_Toc171607409)

[Configuration 4](#_Toc171607410)

[Frontend 6](#_Toc171607411)

[Backend 7](#_Toc171607412)

[Mail sending with google 8](#_Toc171607413)

[Configuration 8](#_Toc171607414)

[Backend 8](#_Toc171607415)

[Pusher 9](#_Toc171607416)

[Configuration 9](#_Toc171607417)

[Frontend 9](#_Toc171607418)

[Backend 9](#_Toc171607419)

[Vue JS Dependency Injection 10](#_Toc171607420)

[1ère façon de faire : 10](#_Toc171607421)

[2ème façon de faire 10](#_Toc171607422)

[Cypress 12](#_Toc171607423)

[Vitest 13](#_Toc171607424)

[Composant générique 14](#_Toc171607425)

# Introduction

Dans ce document, vous trouverez diverses informations pour débuter avec divers outils. Vous devrez aussi garder en tête que les technologies utilisées sont Vue.JS 3 pour le frontend et DotNet pour le backend.

Dans un premier temps, je vous parlerai d’une façon d’implémenter l’inscription et la connexion avec Google aussi bien dans le frontend que le backend.

Ensuite, je vous expliquerai comment intégrer l’envoi de mail dans votre backend.

Et pour finir, je décrirais la manière dont j’ai implémenté « Pusher » pour l’envoi de messages entre plusieurs clients, aussi bien dans le frontend que le backend.

# Authentification Google

Dans cette partie, vous trouverez les informations nécessaires à l’authentification de l’utilisateur via Google, que ce soit pour la connexion ou bien l’inscription.

## Configuration

Avant toute chose, vous devez vous rendre dans votre console Google (et vous créer un compte si ce n’est pas déjà fait) : <https://console.cloud.google.com>

Maintenant, vous pouvez suivre ces étapes :

1. Créez un nouveau projet

Une image contenant capture d’écran, texte, Police, logiciel

Description générée automatiquement

Une image contenant texte, Police, ligne, nombre

Description générée automatiquement

N’oubliez pas de lui donner un nom qui vous permettra de le retrouver facilement.

1. Allez sur la page du projet
2. Configurez l’écran de consentement
   1. Pour ce faire, allez dans « API et services »🡺 « écran de consentement OAuth »
   2. Remplissez les champs obligatoires (vous devrez choisir « external » comme « User Type »)
   3. Attention : il est impératif de sélectionner ces champs d’application non sensibles

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

1. Créez des identifiants OAuth 2.0
   1. Pour ce faire, allez dans « Api et Services » 🡺 « Identifiants » 🡺 Sélectionnez le bouton « Créer des identifiants »
   2. Pour les urls « Origines JavaScript autorisées », entrez les urls de votre serveur de développement front end
   3. Même chose pour les « Uri de redirection autorisées »

## Frontend

Pour le frontend, j’utilise Vue.JS 3 et une librairie « vue3-google-signin » pour permettre l’authentification via Google.

1. Installez la librairie « vue3-google-signin »
2. Retournez dans la console Google et copier l’id client de celui que vous venez de créer
3. Ajoutez ceci dans votre « main.ts »



Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement

1. Collez l’id client que vous avez copié à la place de « YOUR-CLIENT-ID »
2. Ajoutez ceci là où vous voulez permettre à l’utilisateur de se connecter avec google





Une image contenant texte, capture d’écran, Police, ligne

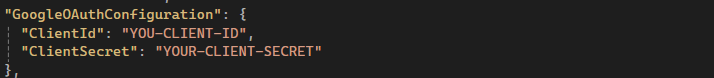
Description générée automatiquement

1. Maintenant, vous pouvez utiliser le token Google comme vous voulez. Le token vous servira à récupérer des informations sur l’utilisateur dans votre backend.

## Backend

L’api est réalisée avec DotNet 8.0.

1. Installez les packages NuGet suivants :
   1. Google.Apis.PeopleService.v1
   2. Google.Apis.Auth
   3. Microsoft.AspNetCore.Authentication.Google
   4. Microsoft.AspNetCore.Authentication.JwtBearer
2. Modifiez votre « appsettings.json » pour avoir cette partie-ci dedans :



1. Ajoutez cette ligne dans votre « Program.cs »

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

1. Vous n’avez plus qu’à ajouter ceci là où vous recevez votre token

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

# L’envoi de mail avec google

## Configuration

Pour pouvoir envoyer un mail depuis votre backend, vous aller devoir créer un nouveau mot de passe d’application comme suit :

1. Aller sur la page de votre compte google
2. Aller dans l’onglet “Sécurité”
3. Activer l’authentification en 2 étapes, si vous ne l’avez pas déjà fait
4. Aller dans « Vérification en 2 étapes »
5. Maintenant plus qu’à naviguer juste qu’à

Une image contenant texte, reçu, Police, capture d’écran

Description générée automatiquement

1. Pour ajouter un mot de passe d’applications il vous suffit de remplir le formulaire qui s’affiche une fois que vous avez cliqué sur la flèche

## Backend

Vous devez simplement ajouter dans le code où vous souhaitez cet extrait de code :

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

# Pusher

## Configuration

Tout d’abord, vous devez vous rendre ici : [pusher.com](https://pusher.com/); et, si vous n’avez pas de compte, vous créez un compte.

Une fois que c’est fait, vous n’avez plus qu’à suivre ces étapes :

* Allez sur votre dashboard : <https://dashboard.pusher.com/>
* Dans vos « Channel », cliquez sur « Manage »

Une image contenant texte, capture d’écran, Police, Graphique

Description générée automatiquement

* Cliquez sur « Create app »

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

* Remplissez le formulaire en y mettant les informations que vous souhaitez.
* Pour finir, vous pouvez cliquer sur « Create app »

## Frontend

* Pour commencer, vous pouvez ajouter la librairie « pusher-js » à votre projet
* Ensuite, dans le composant dans lequel vous voulez recevoir les messages vous pouvez simplement ajouter :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## Backend

* Tout d’abord, il vous faut ajouter pusher à votre projet. Il faut simplement ajouter le package PusherServer
* Ensuite, vous pouvez utiliser le code ci-dessous :

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Dans ce code, on crée une instance de Pusher à laquelle on va donner les informations qui nous ont été fournies lorsque l’application a été créée sur <pusher.com>.

Et pour envoyer le message, il nous suffit d’appeler la méthode « TriggerAsync » en lui fournissant :

* le nom du channel sur lequel envoyer le message
* le nom de l’évènement qui est lancé
* les informations à envoyer sur le channel (dans notre cas, les informations du message qui est envoyé).

Attention à ce que le nom du channel et de l’évènement soient les mêmes dans le frontend et le backend. Sinon, vous ne verrez jamais les messages arriver.

# L’injection de dépendance avec Vue.JS

Avec Vue JS 3 (je ne sais pas pour les autres), il est possible de faire quelque chose qui ressemble à de l’injection de dépendances.

Pour ce faire, vous pouvez le faire de deux manières différentes.

## 1ère façon de faire :

Nous allons voir ici comment le faire via les propriétés des composants. Pour « injecter » on va utiliser la propriété provide tandis que pour « récupérer » on va utiliser inject.

Cette méthode-ci est plus utilisée si vous voulez cibler votre injection au sein des enfants d’un composant.

Nous avons provide qui prend un objet comme valeur. Les clés de cet objet sont le nom de l’injection (celui que vous utiliserez plus tard), et la valeur est ce que vous voulez injecter.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

(L’opération ici se fait dans le fichier « App.vue », car je souhaite pouvoir accéder aux élément injecté partout dans mon application)

La propriété ‘key’ de l’objet donné dans ‘provide’ sera celle qui identifiera notre injection et que nous réutiliserons.

Ensuite, nous avons inject qui prend un tableau et dont chacune des entrées est une chaine de caractères représentant une des clés de la propriété provide du composant parent.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Maintenant, nous pouvons utiliser la valeur qui a été injectée comme ceci :

Une image contenant Police, texte, capture d’écran, Graphique

Description générée automatiquement

## 2ème façon de faire

Nous n’allons plus le faire via des propriétés de composants, mais via des appels de méthodes. Il nous suffit d’utiliser des méthodes du même nom que les propriétés mentionnées avant et elles ont le même usage.

Dans votre fichier *main.ts* vous pouvez simplement ajouter cette ligne :



Dans cet exemple, on injecte une chaine de caractères contenant la valeur ‘value’ et qui est accessible grâce à la clé ‘key’.

Pour vous faciliter la vie, vous pouvez aussi enregistrer vos clés dans un fichier à part, ce qui vous permettra de plus facilement les retrouver.

Maintenant, pour récupérer la valeur, il suffit de faire ceci :



Grâce à l’appel de la méthode ‘inject’ on demande à récupérer la valeur liée à la clé ‘key’ et elle est convertie en ‘string’, on n’a donc pas besoin de faire un cast manuellement.

N’oubliez pas d’ajouter cet import pour pouvoir utiliser la méthode ‘inject’ :

