MINISTÈRE
DE L'ÉCONOMIE,
DES FINANCES
ET DE LA RELANCE
Liberté
Égalité
Fraternité

Agence pour l'informatique financière de l'État

12/04/2024

Guide de migration Captchétat V1 vers V2

CAPTCHA

Destinataires :

- ***Utilisateurs et intégrateur du CAPTCHA souhaitant migrer de la V1 à la V2***

Historique

DATE	*PRENOM NOM*	*STATUT / SUIVI DES MODIFICATIONS*
06/06/2024	*LOT 4*	*V1.2*
06/06/2024	*LOT 4*	*Ajout de la localisation des lib front*

10, Rue du Centre
93160 NOISY-LE-GRAND
Tél : 01 57 33 95 87
https://aife.economie.gouv.fr

Objectif du document	***Description du document*** ***Guide de migration du Captchétat v1 vers le Captchétat v2***
Mots clefs	***Captcha, API***
Résumé	***Guide de migration du Captchétat v1 vers le Captchétat v2***

Sommaire

1 Synoptique

Un captcha permet de déterminer si l'utilisateur est un être humain ou un robot, à travers un test visuel ou auditif. Il permet de contrer l'utilisation de robots, qui pourraient accroître exponentiellement le nombre de requêtes (spam), ce qui pourrait provoquer une surchage sur le serveur (DDOS). C'est dans cette dynamique que l'AIFE a mis en place sa propre solution de captcha. Le présent document décrit le mode opératoire d'implémentation du captcha de l'AIFE.

2 Prérequis

Voici les prérequis techniques requis pour l'implémentation du captcha :

NAVIGATEUR	ANGULAR-CLI	ANGULARJS	REACT
Chrome 49+	Angular 8	Angular 1.8.3	React 18
Edge 20+	Angular 12-17		
Firefox 52+			
IE 8+			
Opera 36+			
Safari (OSX) 5+			
Safari (iOS 6+)			

Les différentes librairies front sont accessibles aux url suivantes :

- ***Angular 8 :*** *https://static.piste.gouv.fr/captchEtat/libraries/captchetat-angularv8-1.0.2.tgz*
- ***Angular 12-17 :*** *https://static.piste.gouv.fr/captchEtat/libraries/captchetat-angular-1.0.0.tgz*
- ***AngularJS :*** *https://static.piste.gouv.fr/captchEtat/libraries/captchetat-angularjs-1.0.0.tgz*
- ***React :*** *https://static.piste.gouv.fr/captchEtat/libraries/captchetat-react-1.0.0.tgz*
- ***Javascript :*** *https://static.piste.gouv.fr/captchEtat/libraries/captchetat-js-1.0.0.tgz*

Elles peuvent être récupérées soit via un npm install, soit directement sous forme d'archive aux urls fourni ci-dessus.

En cas d'utilisation d'autres frameworks front non supportés par les librairies fournies ou de difficultés dans l'implémentation de ces dernières, ne pas hésiter à revenir vers les équipes Support.

3 Types de captcha

Les différents types de captcha utilisables ainsi que leurs caractéristiques sont listés ci-dessous :

Valeurs	*Nombre de caractères*	*Type de caractères*	*Niveau de difficulté*
captchaFR	*6-9*	*Alphanumérique*	*NORMAL*
captchaEN	*6-9*	*Alphanumérique*	*NORMAL*
numerique6_7CaptchaFR	*6-7*	*Numérique*	*DIFFICULT*
numerique6_7CaptchaEN	*6-7*	*Numérique*	*DIFFICULT*
alphabetique6_7CaptchaFR	*6-7*	*Alphabétique*	*DIFFICULT*
alphabetique6_7CaptchaEN	*6-7*	*Alphabétique*	*DIFFICULT*
alphanumerique12CaptchaFR	*12*	*Alphanumérique*	*NORMAL*
alphanumerique12CaptchaEN	*12*	*Alphanumérique*	*NORMAL*
alphabetique12CaptchaFR	*12*	*Alphabétique*	*NORMAL*
alphabetique12CaptchaEN	*12*	*Alphabétique*	*NORMAL*
numerique12CaptchaFR	*12*	*Numérique*	*NORMAL*
numerique12CaptchaEN	*12*	*Numérique*	*NORMAL*
alphanumerique6to9LightCaptchaEN	*6-9*	*Alphanumérique*	*LIGHT*
alphanumerique6to9LightCaptchaFR	*6-9*	*Alphanumérique*	*LIGHT*
alphanumerique4to6LightCaptchaEN	*4-6*	*Alphanumérique*	*LIGHT*
alphanumerique4to6LightCaptchaFR	*4-6*	*Alphanumérique*	*LIGHT*

Tableau 1. Valeurs possibles et caractéristiques des types de captcha

4 Modifications front-end

Les bibliothèques front sont disponibles sur le serveur statique https://static.piste.gouv.fr/captchEtat/ (accès libre) dans le répertoire librairies.

4.1 Angular CLI

Par souci de compatibilité avec un maximum de version d'angular, deux bibliothèques sont disponibles : captchetat-angularv8 et captchetat-angular. Captchetat-angular n'est compatible qu'avec les applications à partir d'angular 12, pour les applications basées sur une version inférieure d'angular, il faudra utiliser la bibliothèque captchetat-angularv8.

Dans l'ensemble des étapes suivantes, l'ancien et nouveau code sont surlignés de couleurs différentes (rouge pour l'ancien et violet pour le nouveau). Dans la majorité des cas, il suffit de remplacer le code surligné en rouge par celui surligné en violet.

4.1.1 Modification de la bibliothèque

La première étape consiste à désinstaller ou supprimer la bibliothèque botdetect utilisée précedemment, via la commande suivante :

```
npm uninstall angular- captcha
```

Ensuite, installer le package de la bibliothèque captcha, à récupérer depuis le NGINX via un npm install de la manière suivante (pour Angular 8 ou inférieur, récupérer le package captchetat-angularv8-1.0.2.tgz) :

```
npm i https://static.piste.gouv.fr/captchEtat/libraries/captchetat-angular-1.0.0.tgz
ou
npm i https://static.piste.gouv.fr/captchEtat/libraries/captchetat-angularv8-1.0.2.tgz
```

Un accès à l'url https://static.piste.gouv.fr/captchEtat/ est nécessaire pour la récupération de la bibliothèque.

4.1.2 Remplacement de la dépendance dans l'app.module

Dans votre app.module.ts, remplacer l'import de BotdetectCaptchaModule par celui de CaptchetatAngularModule :

```
import { BotDetectCaptchaModule } from 'angular-captcha';
import { CaptchetatAngularModule } from 'captchetat-angular';
…
@NgModule({
    declarations: [
        …
    ],
    imports: [
        BotDetectCaptchaModule,
        CaptchetatAngularModule,
       …
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule {}
```

4.1.3 Modification du formulaire

Remplacer la balise botdetect-captcha par la balise captchetat, comme suit :

- *Dans la partie HTML :*

```
<botdetect-captcha captchaStyleName={{nomCaptcha}} autoFocusInput="false"></botdetect-captcha >
```

```
<captchetat captchaStyleName={{nomCaptcha}} altImage={{alternativeImage}} urlBackend={{urlBackend}}>
</captchetat>
<input id="captchaFormulaireExtInput" name="captchaFormulaireExtInput" formControlName="captchaCode"
type="text"/>
```

La variable alternativeImage est optionnelle et correspond à une phrase alternative pour la valeur du alt de l'image. Par défaut, si la variable altImage n'est pas renseignée dans la balise, elle prend la valeur 'Recopier le code de sécurité de cette image' ;

La variable urlBackend correspond à 'urlBackend /simple-captcha-endpoint'. Il faut remplacer urlBackend par l'URL pour accéder à votre backend, car nous allons y exposer une API permettant de récupérer le captcha. Si votre backend dessert votre frontend, vous n'êtes pas dans l'obligation de préciser l'url du backend.

La variable nomCaptcha peut prendre les valeurs référencées dans le tableau 1.

- ***Dans la partie typescript de votre formulaire, remplacer l'import de CaptchaComponent par celui de CaptchetatAngularComponent et supprimer le ngOnInit() dédié à l'initialisation du captchaEndpoint de botdetect :***

```
import { Component, ViewChild } from '@angular/core';
import { CaptchaComponent } from 'angular-captcha';
import { CaptchetatAngularComponent } from 'captchetat-angular';
import { YourFormWithCaptchaService } from './your-form-with-captcha.service';

@Component({
  selector: 'your-form-with-captcha',
  standalone: true,
  imports: [],
  providers: [YourFormWithCaptchaService],
  templateUrl: 'your-form-with-captcha.component.html',
  styleUrl: 'your-form-with-captcha.component.css'
})
export class YourFormWithCaptchaComponent {
  @ViewChild(CaptchaComponent, { static: true) captchaComponent: CaptchaComponent;
  @ViewChild(CaptchetatAngularComponent, { static: true) captchetatComponent:
CaptchetatAngularComponent;

  constructor(private yourFormWithCaptchetatAngularService: YourFormWithCaptchetatAngularService) {}

  urlBackend: string = 'chemin/vers/votre/backend';
  nomCaptcha: string = 'captchaFR';
  altImage: string = 'alternative alt pour image';

  ngOnInit(): void {
    this.captchetatComponent.captchetatComponent =
      'URL_BACKEND/simple-captcha-endpoint'
  }

}
```

4.1.4 Modification de la partie validation

L'objet Captcha est légèrement modifié par rapport à la V1, il faut remplacer l'attribut «id » par « uuid » (cela n'est pas absolument nécessaire, mais c'est par soucis de cohérence avec l'objet json envoyé au endpoint de validation par la partie backend, cf 5):

- ***Création de la classe captcha :***

```
export class Captcha {
    private uuid: string;
    private code: string;

    constructor(uuid: string, code: string){
        this.uuid = uuid;
        this.code = code;
    }
}
```

- ***L'instanciation de l'objet captcha dans la méthode sendValidation() partie typescript du formulaire change légèrement pour ce qui est de la récupération de l'uuid et devient :***

```
let captcha = new Captcha(this.captchaComponent.captchaId,
this.myFormWithCaptchetat.get('captchaCode').value as string);
let captcha = new Captcha(this.captchetatComponent.getIdCaptcha(),
this.myFormWithCaptchetat.get('captchaCode').value as string);
```

- ***La transmission au service permettant de communiquer avec le backend, ainsi que la gestion possible des erreurs ne nécessitent pas de modification particulière.***

4.2 Angular JS

Dans l'ensemble des étapes suivantes, l'ancien et nouveau code sont surlignés de couleurs différentes (rouge pour l'ancien et violet pour le nouveau). Dans la majorité des cas, il suffit de remplacer le code surligné en rouge par celui surligné en violet.

4.2.1 Modification de la bibliothèque

La première étape consiste à désinstaller ou supprimer la bibliothèque botdetect utilisée précedemment, via la commande suivante :

```
npm uninstall angularjs- captcha
```

Ensuite, installer le package de la bibliothèque captcha, à récupérer depuis le NGINX via un npm install de la manière suivante :

```
npm i https://static.piste.gouv.fr/captchEtat/libraries/captchetat-angularjs-1.0.0.tgz
```

Un accès à l'url https://static.piste.gouv.fr/captchEtat/ est nécessaire pour la récupération de la bibliothèque.

4.2.2 Remplacement du script dans la page d'index

Remplacer l'appel au script botdetect par un appel au script captchetat-angularjs dans votre index.html :

```
<script src="node_modules/angularjs-captcha/dist/angularjs-captcha.min.js"></script>
<script src="node_modules/captchetat-angularjs/dist/captchetat-angularjs.js"></script>
```

4.2.3 Remplacement du captcha dans le formulaire

Remplacer la balise botdetect-captcha par la balise captchetat comme suit :

- *Dans la partie HTML :*

```
<botdetect-captcha captcha-style-name={{nomCaptcha}} autoFocusInput="false"></botdetect-captcha>
<captchetat captcha-style-name={{nomCaptcha}} alt-image={{alternativeImage}} url-
backend={{urlBackend}}></captchetat>
<input id="captchaFormulaireExtInput" name="captchaFormulaireExtInput" formControlName="captchaCode"
type="text"/>
```

La variable alternativeImage est optionnelle et correspond à une phrase alternative pour la valeur du alt de l'image. Par défaut, si la variable altImage n'est pas renseignée dans la balise, elle prend la valeur 'Recopier le code de sécurité de cette image' ;

La variable urlBackend correspond à 'urlBackend /api/simple-captcha-endpoint'. Il faut remplacer URL_BACKEND par l'URL pour accéder à votre backend, car nous allons y exposer une API permettant de récupérer le captcha. Si votre backend dessert votre frontend, vous n'êtes pas dans l'obligation de préciser l'url du backend.

La variable nomCaptcha peut prendre les valeurs référencées dans le tableau 1.

Dans la partie javascript de votre formulaire, remplacer l'import du module BotDetectCaptcha, par celui de CaptchEtatAngularJS, et supprimer la partie servant à initialiser la config de botdetect :

```
var app = angular.module('myApp', ['BotDetectCaptcha','CaptchEtatAngularJS'])
    .controller('FormController', function($scope) {
        $scope.formData = {};

        $scope.submitForm = function() {
            $scope.submitted = true;
        };

       captchaSettingsProvider.setSettings({
         captchaEndpoint:
           'URL_BACKEND/simple-captcha-endpoint'
       });
});
```

4.2.4 Modification de la partie validation

Pour la validation du captcha, plus besoin de passer par l'instanciation d'un new Captcha(). Il suffit donc de supprimer cette partie, et de la remplacer par la partie surlignée en violet. On récupère directement depuis l'html les valeurs du code fournit par l'utilisateur ainsi que l'uuid associé au captcha :

```
app.controller('YourFormWithCaptchaController',
function ($scope, $http, $window, Captcha, CaptchEtatService) {

        // Fonction appelée lors de la soumission du formulaire
        $scope.validate = function () {

        // Création d'une nouvelle instance captcha
        var captcha = new Captcha();

        // Récupération du code captcha saisi par l'utilisateur
        var userEnteredCaptchaCode = captcha.getUserEnterdCaptchaCode();
        var userEnteredCaptchaCode = document.getElementById("captchaCode").value;

        // Récupération de l'id du captcha généré par l'API
        var captchaId = captcha.getCaptchaId();
        var captchaId = document.getElementById("captchetat-uuid").value;

        //Objet envoyé au backend pour valider le captcha contenant l'id et le code du cap tcha
        var postData = {
            uuid: userEnteredCaptchaCode,
            code: captchaId
        };

        // Envoi au backend pour validation du formulaire incluant le captcha
        $http({
            headers: { 'Content-Type': 'application/json; charset=utf-8' },
            method: 'POST',
            url: 'urlBackend/validationFormulaire',
            data: postData
        })
            .then(function (response) {
                if (response.data.success == false) {
                // La validation du captcha est KO, régénération de l'image
                   captcha.reloadImage();
                   CaptchEtatService.reloadCaptcha();
                // TODO: Afficher un message d'erreur
                } else {
                    // La validation du captcha est OK.
                }
                }, function (error) {
                    throw new Error(error.data);
            });
    };
    });
```

4.3 Javascript

Dans l'ensemble des étapes suivantes, l'ancien et nouveau code sont surlignés de couleurs différentes (rouge pour l'ancien et violet pour le nouveau). Dans la majorité des cas, il suffit de remplacer le code surligné en rouge par celui surligné en violet.

4.3.1 Modification de la bibliothèque captcha

La première étape est de supprimer la bibliothèque botdetect, soit via un npm uninstall (si cette dernière a été ajoutée via npm install), soit en supprimant manuellement le fichier jquery-captcha.min.js de votre répertoire contenant les sources externes.

La bibliothèque js peut être récupérée depuis le NGINX sous forme d'archive. Il faut en extraire le fichier captchetat-js.js et l'ajouter au répertoire ASSETS_JS (ou tout autre répertoire contenant vos sources externes).

Il est également possible de la récupérer via un npm install comme suit :

```
npm i https://static.piste.gouv.fr/captchEtat/libraries/captchetat-js-1.0.0.tgz
```

Un accès à l'url https://static.piste.gouv.fr/captchEtat/ est nécessaire pour la récupération de la bibliothèque.

4.3.2 Appel du plugin captcha depuis le frontend

Côté frontend, dans la page du formulaire, remplacer l'appel au script botedetect par l'appel au script captchetat:

```
<script src="ASSETS_JS/jquery-captcha.min.js"></script>
<script src="ASSETS_JS/captchetat-js.js"></script>
```

Remplacer 'ASSETS_JS' par le chemin d'insertion du script.

4.3.3 Remplacement dans le formulaire

Dans la partie HTML du formulaire, remplacer la balise d'id « botdetect-captcha » par la balise d'id « captchetat », comme suit :

```
<div id="botdetect-captcha" data-captchaStyleName="nomCaptcha"></div>
<div id="captchetat" captchaStyleName="nomCaptcha" altImage="alternativeImage"
urlBackend="urlBackend"></div>
<input type="text" placeholder="SAISIR LE CAPTCHA" id="captchaFormulaireExtInput"/>
```

La variable alternativeImage est optionnelle et correspond à une phrase alternative pour la valeur du alt de l'image. Par défaut, si la variable altImage n'est pas renseignée dans la balise, elle prend la valeur 'Recopier le code de sécurité de cette image' ;

La variable urlBackend correspond à 'urlBackend /simple-captcha-endpoint'. Il faut remplacer urlBackend par l'URL pour accéder à votre backend, car nous allons y exposer une API permettant de récupérer le captcha. Si votre backend dessert votre frontend, vous n'êtes pas dans l'obligation de préciser l'url du backend.

La variable nomCaptcha peut prendre les valeurs référencées dans le tableau 1.

4.3.4 Modification de la partie validation

La manière de récupérer le code saisi et l'id du captcha est légèrement différente. Remplacer dans votre js comme suit :

```
$('#myFormWithCaptchetat').submit(function(event) {
  // Le code entré par l'utilisateur récupéré en backend
  var captchaCode = captcha.getUserEnterdCaptchaCode();
  var captchaCode = document.getElementById('captchaCode').value;

  // id du captcha que l'utilisateur a tenté de résoudre
  var captchaUuid = captcha.getCaptchaId();
  var captchaUuid = document.getElementById('captchetat-uuid').value;

  var postData = { uuid: captchaUuid, code: captchaCode };
```

```
// Envoi des données du captcha côté backend
$.ajax({
  method: 'POST',
  url: 'urlBackend/validationFormulaire',
  dataType: 'json',
  contentType: 'application/json; charset=utf-8',
  data: JSON.stringify(postData),
   success: function(response) {
      •                    if (response.success == false) {
      •                       // La validation a échoué. Le captcha est rechargé
      •                       captcha.reloadImage();
      •                       window.captchetatComponentModule.refreshCaptcha();
      •                    } else {
      •                       // le captcha a été correctement rempli. Validation côté front.
                  }
  }
});

  event.preventDefault();
});
      •
```

L'appel à la validation se fait maintenant avec la fonction window.captchetatComponentModule.refreshCaptcha(), comme le montre cet exemple.

4.4 React

Dans l'ensemble des étapes suivantes, l'ancien et nouveau code sont surlignés de couleurs différentes (rouge pour l'ancien et violet pour le nouveau). Dans la majorité des cas, il suffit de remplacer le code surligné en rouge par celui surligné en violet.

4.4.1 Modification de la bibliothèque captcha

La première étape consiste à désinstaller ou supprimer la bibliothèque botdetect utilisée précedemment, via la commande suivante :

```
npm uninstall reactjs-captcha
```

Ensuite, installer le package de la bibliothèque captcha, à récupérer depuis le NGINX via un npm install de la manière suivante :

```
npm i https://static.piste.gouv.fr/captchEtat/libraries/captchetat-react-1.0.0.tgz
```

Un accès à l'url https://static.piste.gouv.fr/captchEtat/ est nécessaire pour la récupération de la bibliothèque.

4.4.2 Remplacement du captcha dans le formulaire

Ajouter le captcha dans votre formulaire de cette manière :

```
import React from "react";
import axios from 'axios';
import { Captcha, captchaSettings } from 'reactjs-captcha;
import { CaptchEtat } from 'captchetat-react';

class YourFormWithCaptcha extends React.Component {
   const urlBackend='chemin/vers/votre/back';
   const captchaStyleName = "captchaFR";
   const alternativeImage = "Alternative image";

  const captchetatRef = useRef(null);

  const handleChange = (event) => {
    setValues({...values, [event.target.name]: event.target.value});
  }
  // Fonction permettant de mettre à jour l'uuid du captcha à envoyer pour la validation
  const onCaptchaChange = (captcha) => {
    setValues({...values, captchaId: captcha.uuid})
  }
      […]
render() {
return (
    <div className="App">
      <form id="yourFormWithCaptchaForm" onSubmit={handleSubmit}>
      <div className="form-group">
        <Captcha captchaStyleName={captchaStyleName} ref={(captcha) => {this.captcha = captcha}}/>
        <CaptchEtat urlBackend={urlBackend} captchaStyleName={captchaStyleName}
altImage={alternativeImage} onChange={onCaptchaChange} className="captchetat" ref={captchetatRef}/>
        <input name="captchaCode" type="text" onChange={handleChange} placeholder="Saisir le captcha"
className="captchetat-response"/>
        <input type="submit" value="Submit" />
```

```
      </div>
     </form>
    </div>
  )}
}
export default YourFormWithCaptcha;
```

La variable alternativeImage est optionnelle et correspond à une phrase alternative pour la valeur du alt de l'image. Par défaut, si la variable altImage n'est pas renseignée dans la balise, elle prend la valeur 'Recopier le code de sécurité de cette image' ;

La variable urlBackend correspond à 'urlBackend /simple-captcha-endpoint'. Il faut remplacer urlBackend par l'URL pour accéder à votre backend, car nous allons y exposer une API permettant de récupérer le captcha. Si votre backend dessert votre frontend, vous n'êtes pas dans l'obligation de préciser l'url du backend.

La variable nomCaptcha peut prendre les valeurs référencées dans le tableau 1.

Le hook [values, setValues] sert à mettre à jour les champs du formulaire à récupérer et donc le captchaCode et l'uuid à envoyer au backend pour la validation.

La ref captchetatRef sert à appeler la fonction reloadImage(), pour pouvoir recharger le captcha si la réponse est fausse.

La fonction handleChange sert à mettre à jour la valeur du code à envoyer au backend pour validation lorsque l'utilisateur interagit avec l'input en question (ici name= « captchaCode »).

4.4.3 Ajout de la validation du captcha

La validation du captcha se fait à la soumission du formulaire dans la partie backend de l'application. Il faut donc transmettre le captcha à votre backend de la même manière que vous transmettez les autres champs du formulaire. Le traitement préconisé dans le guide d'implémentation de la v1 peut être intégralement remplacer par le suivant, qui utilise le hook « values » décrit précédemment pour mettre à jour le JSON à envoyer au backend :

```
// Fonction appelée lors de la soumission du formulaire
const handleSubmit = async (event) => {
    event.preventDefault();
    try {
    // Envoi au backend pour validation du formulaire incluant le captcha
      const response = await axios.post(`/api/submit`, values);
      if (response.data.success == false) {
                // La validation du captcha est KO, régénération de l'image
                captchetatRef.current?.reloadImage();
                // TODO: Afficher un message d'erreur
            } else {
                // La validation du captcha est OK.
            }
    } catch (exception) {
      // Traitement de l'exception
    } ;

    let self = this;
    event.preventDefault();
}
```

La partie front-end est terminée vous pouvez passer directement à la partie backend.

5 Implémentation captcha back-end

Pour pouvoir accéder aux API captcha exposées dans PISTE, les actions à réaliser sont exactement les mêmes que pour la v1 :

- ***Récupération du jeton Oauth 2.0***
- ***Exposer une API permettant de récupérer le captcha par un appel API vers PISTE.***
- ***Valider le captcha par un appel API vers PISTE.***

5.1 Récupération du jeton Oauth 2.0

La récupération du jeton Oauth 2.0 se déroule exactement de la même manière qu'auparavant (voir guide d'implémentation si besoin).

5.2 Récupération du captcha

Pour la récupération du captcha, nous allons mettre à disposition de notre front-end l'api : GET /simple-captchaendpoint, comme pour la v1. La seule différence est l'url du endpoint:

- ***Appel GET à /piste/captcha/simple-captcha-endpoint pour la v1 ;***
- ***Appel GET à /piste/captchetat/v2/simple-captcha-endpoint pour la v2.***

Cette API va récupérer les headers des différents appels générés par la bibliothèque front-end et renvoyer ces appels avec les headers correspondants vers l'API Piste, en ajoutant le jeton Oauth dans les headers. Voici un schéma récapitulatif des échanges entre le front-end, le back-end et Piste pour cette API :

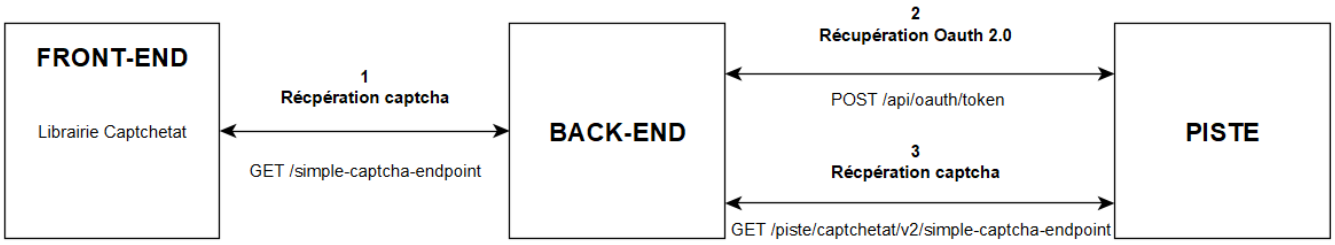

Figure 1. Échanges pour la récupération du Captcha en v2

Lors de la récupération du captcha depuis le back-end vers PISTE, il faut ajouter le jeton Oauth 2.0, qui a une durée de vie de 3600 secondes, dans le header de l'appel API. Pour se faire, il faut ajouter dans le header : Authorization: Bearer « access_token ».

Le service captcha ne génère plus directement du code html, mais un json contenant un string base64 de l'image ainsi que l'uuid du captcha associé, ou bien le son sous forme de byte array via un appel à l'url /api/simple-captcha-endpoint?[paramètre]… . Ce sont les bibliothèques front qui se chargent de construire le html du composant avec les données récupérées via l'appel.

Le service captcha v2 ne fait plus appel à des ressources statiques à partir de l'URL https://static.piste.gouv.fr/captchEtat/ , il n'est donc pas nécessaire d' avoir l'accès à l'URL https://static.piste.gouv.fr/captchEtat/* pour que le service catpcha fonctionne correctement depuis un réseau fermé.

En revanche, l'accès au serveur statique est nécessaire pour la récupération des bibliothèques front à implémenter, comme indiqué précédemment.

Pour résumé, voici les différentes étapes pour la récupération du captcha (à partir du schéma précédent) :

1. ***La bibliothèque captcha ajoutée au préalable va générer un appel API : GET /simple-captcha-endpoint avec comme paramètre get et c. Exemple :***
    - ***/simple-captcha-endpoint?get=image&c=captchaFR (appelé au chargement de la page et au clic sur le bouton de rechargement)***

- */simple-captcha-endpoint?get=sound&c=captchaFR&t=63c68f5c-0c71-40d4-a195-2e3761cdd9b1 (appelé au clic sur le bouton pour jouer le son)*

2. *Le back-end va récupérer le jeton Oauth 2.0 (« access_token ») à travers un appel API vers PISTE : GET /api/oauth/token.*
3. *Le back-end va générer des appels API vers PISTE pour récupérer le captcha, en utilisant les paramètres renseignés par la bibliothèque côté front-end et les headers associés, tout en ajoutant l'access token dans les headers (Authorization: Bearer « access_token ») et en modifiant les urls du code html généré par le service captcha.*

Dans le cas où vous implémentez le proxy en Next.js avec la bibliothèque axio, vous avez la possibilité de renvoyer les appels avec les headers de cette manière :

```
axios({
method: 'get',
  url: 'URL_PISTE_AVEC_PARAMÈTRES_RÉCUPERÉS_DEPUIS_LE_FRONT',
responseType: 'stream'
})
  .then(function (response) {
resultatARenvoyer.pipe(response);   });
```

resultatARenvoyer étant la réponse renvoyée vers le frontend.
De cette manière, les headers ainsi que le corps de la réponse sont renvoyés vers le frontend.

5.3 Validation du captcha

A la soumission du formulaire, un appel API est envoyé au back-end pour valider la conformité des données saisies par l'utilisateur. Avant de vérifier ces données, il faut générer, comme avec la v1, un appel vers PISTE pour la validation du captcha en utilisant le jeton Oauth 2.0.
En v2, c'est le body du captcha qui change, le paramètre « id » est à remplacer par « uuid » :

- ***{"id": "identifiant du captcha", "code" : "code saisi par l'utilisateur"} pour la v1 ;***
- ***{"uuid": "identifiant du captcha", "code" : "code saisi par l'utilisateur"} pour la v2.***

Ainsi que l'url cible :

- ***Appel POST à /piste/captcha/valider-captcha pour la v1 ;***
- ***Appel POST à /piste/captchetat/v2/valider-captcha pour la v2.***

Voici un schéma récapitulatif des échanges entre le front-end, le backend et Piste pour cette API :

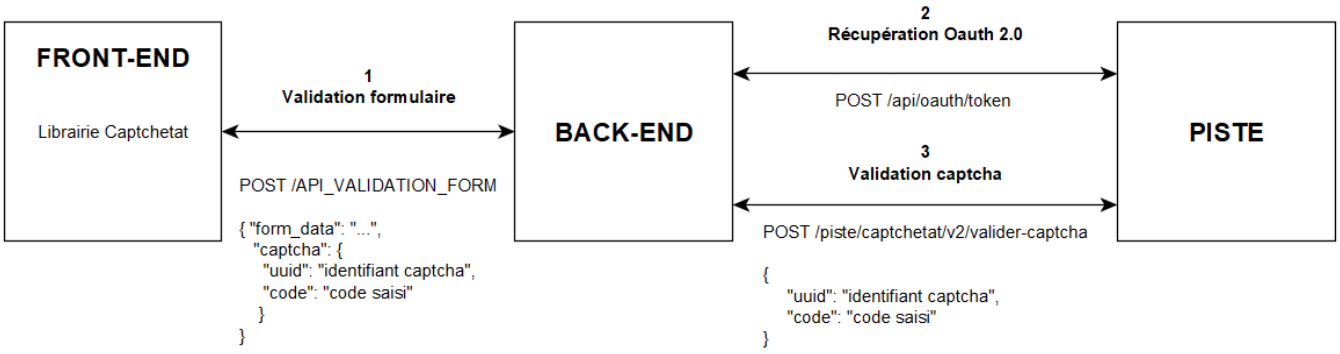

Figure 2. Échanges pour la validation du Captcha en v2

Les étapes pour la validation en v2 sont donc identiques à celles de la v1 :

1. *Lors de la soumission du formulaire, un appel vers le back-end depuis le front-end est généré pour valider les données saisies par l'utilisateur. Parmi ces données, se trouve l'identifiant du captcha (généré par la bibliothèque) et le code saisi par l'utilisateur.*
2. *Avant la validation des champs du formulaire, le back-end va passer à validation le captcha. Pour se faire, il va tout d'abord récupérer le jeton Oauth 2.0 pour pouvoir consommer les API captcha exposées sur PISTE. Puis, il va générer un appel API POST /piste/captchetat/v2/valider-captcha avec pour corps de requête :*

```
{
  "uuid" : "Identifiant du captcha",
  "code" : "Code du captcha saisi par l'utilisateur"
}
```

Dans le cas où le captcha est valide, l'appel renverra « true » et la validation des champs du formulaire pourra débuter.
Dans le cas où le captcha est invalide, l'appel renverra « false » et la validation des champs du formulaire est bloquée. Un message d'erreur est renvoyé au front-end qui régénèrera un nouveau captcha.

5.4 Récupération des informations du Captcha

En complément des méthodes de récupération et de validation, une méthode GET permettant de récupérer les informations du CAPTCHA généré est, comme avec la v1, mise à disposition. Cette méthode permet à partir de l'Id d'un code captcha généré de récupérer les informations sur le type (numérique, alphabétique ou alphanumérique) et le nombre de caractères qui le composent. Elle fonctionne exactement de la même façon qu'en v1 : GET /captcha/{id}/code/infos. Avec {id} : l'Id du code captcha généré.

6 Evolutions et améliorations techniques

Plusieurs évolutions et améliorations techniques ont été apportées à Captchétat v2 par rapport à la v1, ces dernières sont synthétisées ci-après.

6.1 Montée de version Java

La réalisation de la v2 du Captchétat a été l'occasion d'effectuer une montée de version du jdk utilisé par le backend, de java 11 à java 17, et par conséquent de passer de spring-boot 2.6 à spring-boot 3.2 également.

Le socle technique backend est donc plus récent, plus facile à maintenir et présente beaucoup moins de vulnérabilité que celui de la v1. En effet, la v1 présente 70 vulnérabilités selon le rapport dependency-check basé sur les normes de l'OWASP, contre uniquement 4 vulnérabilités en v2 selon le même rapport.

6.2 Abandon de Botdetect

Le sujet principal de la réécriture du Captchétat était l'abandon de l'utilisation de la bibliothèque sous licence Botdetect pour un passage à une solution en opensource intégral.

Côté backend, le fonctionnement via Botdetect a été remplacé par l'utilisation de la bibliothèque opensource Nanocaptcha (v1.5), qui permet la génération à la fois de captcha audio et image.
La librairie a été surchargée par nos soins pour étoffer le choix d'images et de sons proposés et ainsi répondre aux exigences de l'AIFE quant à la solution mise à disposition.

Côté frontend, des bibliothèques ont été implémentées dans 4 frameworks différents (Angular, AngularJS, ReactJS, JS natif), pour couvrir un panel d'application au moins aussi important que celui couvert par les bibliothèques botdetect.

6.3 Amélioration du nombre d'appel

Le nombre d'appel API suite à un appel du service par défaut (appel déclenché par un accès à une page web sur lequel le composant CaptchEtat est implémenté) a été drastiquement réduit entre la v1 et la v2. En effet, ce dernier a été réduit de 8 en v1 à 1 seulement en v2. Les schémas suivants synthétisent le déroulement des appels selon la version :

Figure 3. Fonctionnement appel par défaut Captchétat v1

Figure 4. Fonctionnement appel par défaut Captchétat v2

En v2, il n'y a plus d'appel au serveur de ressources statiques, toutes les sources sont fournies directement via les librairies (les icônes des boutons sont des SVG récupérés depuis le design system de l'Etat).
Un seul appel au backend est déclenché au chargement du composant Captchetat pour récupérer le json permettant de charger l'image.

Pour ce qui est du refresh, de la génération du son et de la validation, le fonctionnement est similaire à celui de la v1, un seul appel est déclenché vers le endpoint concerné.

6.4 Accessibilité

En termes d'accessibilité, les bibliothèques front Captchetat v2 intègrent des améliorations permettant de rendre le composant aux normes d'accessibilité RGAA, dont notamment :

- ***Le contraste minimum de l'image généré ne peut pas dépasser le minimum autorisé par le RGAA ;***
- ***Tous les composant sont atteignables au clavier (avec prise de focus visible et ordre de tabulation cohérent) ;***
- ***La perte de focus après 1 écoute ou un rafraichissement constaté avec la v1 a été corrigé par la v2.***