

# Projet Web des Objets

Cahier des charges/Dossier de  
conception

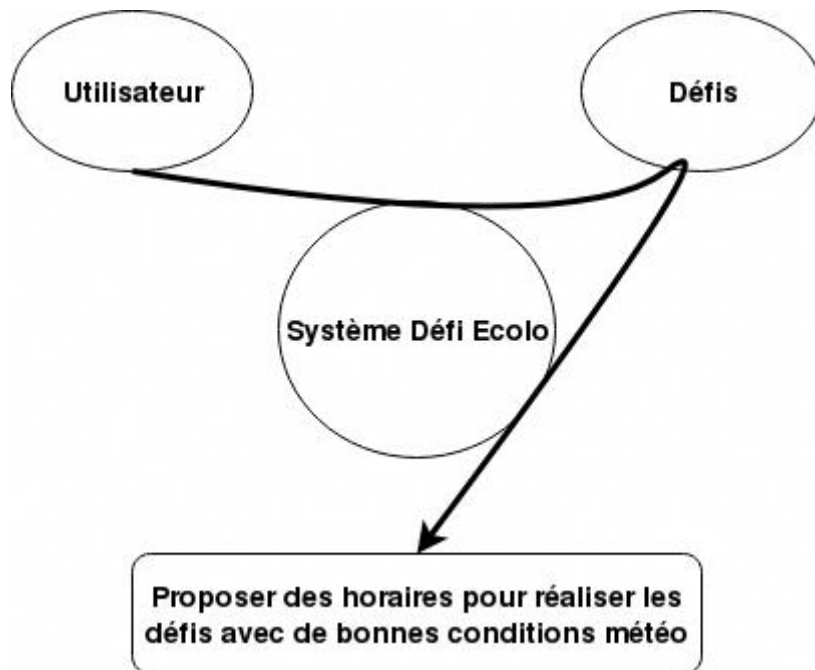
## Table des matières

Cahier des charges .....	3
Expression fonctionnelle du besoin .....	3
Description technique du besoin .....	4
Front-End .....	4
Back-End .....	4
Dossier de conception .....	6
Cas d'utilisation .....	6
Architecture client-serveur .....	7
Implémentation Front-End .....	8
Partie IHM .....	8
Interface avec le serveur : .....	12
Implémentation Back-End .....	13
Diagramme de classe PHP .....	13
Interface avec les clients .....	14
Interface OpenWeatherMap .....	14

## Cahier des charges

### Expression fonctionnelle du besoin

Le système doit permettre aux utilisateurs de proposer des lieux de rendez-vous pour réaliser des défis écolos et de mettre en relation les utilisateurs disponibles sur des créneaux horaires pour planifier ces défis.



*Analyse fonctionnelle du besoin du système*

Ci-dessous les différentes fonctionnalités identifiées du système pour répondre à ce besoin :

- L'utilisateur doit pouvoir s'authentifier afin de pouvoir proposer un défi
- L'utilisateur authentifié doit pouvoir proposer des lieux de rendez-vous pour un défi
- L'utilisateur doit pouvoir renseigner au système ses disponibilités
- Le système doit pouvoir, sur un lieu défini par un utilisateur, proposer des créneaux horaires où une action peut être planifiée en fonction de la météo (qui doit être qualifiée de « bonne »)
- Les utilisateurs doivent pouvoir être notifiés de la présence d'un défi sur un créneau qu'ils ont renseigné disponible
- Ils doivent pouvoir accepter ou refuser le défi
- Le système doit pouvoir en cas de changement des prévisions météorologiques (de « bonne » vers un autre statut), notifier les utilisateurs inscrits sur le créneau concerné du changement de météo et donc de l'annulation du défi

## Description technique du besoin

Dans le cadre de la conception de la maquette, le système doit être accessible à tout moment et à partir de n'importe quelle plateforme pour un utilisateur ayant accès à Internet et prendra la forme d'une application Web.

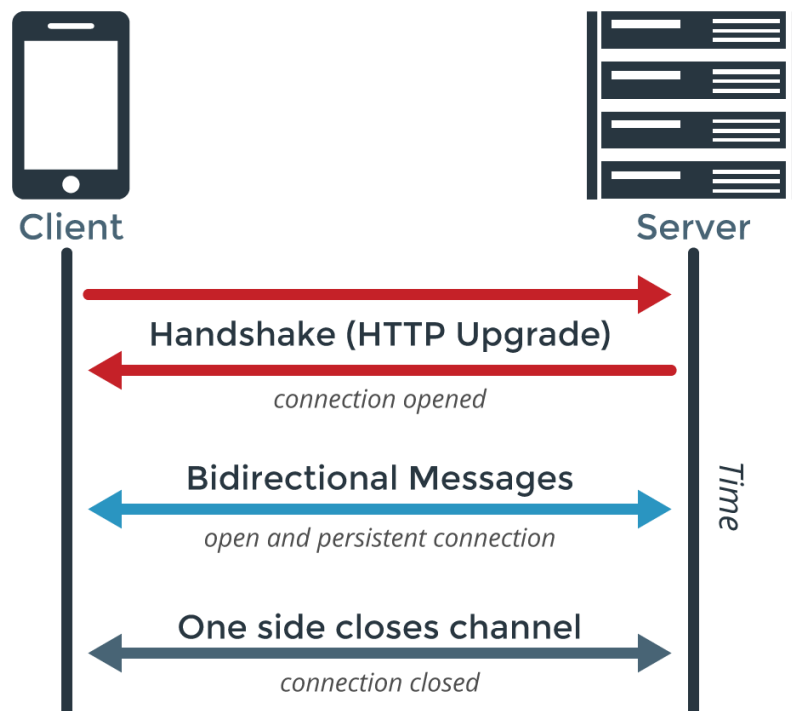
### Front-End

La vue des différentes pages de l'application doit être dynamique afin de répondre au besoin de notification de l'utilisateur à la suite de nouvelles informations reçues par le serveur. Les pages doivent pouvoir afficher ces notifications et les nouvelles informations sans que l'utilisateur ait besoin de les rafraîchir. Des communications doivent donc être initiées côté serveur vers les clients.

Aussi, afin de garantir une expérience utilisateur simple et agréable, un minimum de page sera utilisé sur l'application.

### Back-End

Pour répondre à ces besoins côté client et étant donné la forte interactivité et les multiples connexions http qu'implique le fonctionnement de l'application (e.g notification aux utilisateurs de défis disponibles aux créneaux renseignés), la solution des Websocket a été choisie afin de créer un canal de communication full-duplex permettant l'interactivité client-serveur.



Concernant la récupération de la météo, une source externe sera utilisée à savoir OpenWeatherMap via son API.

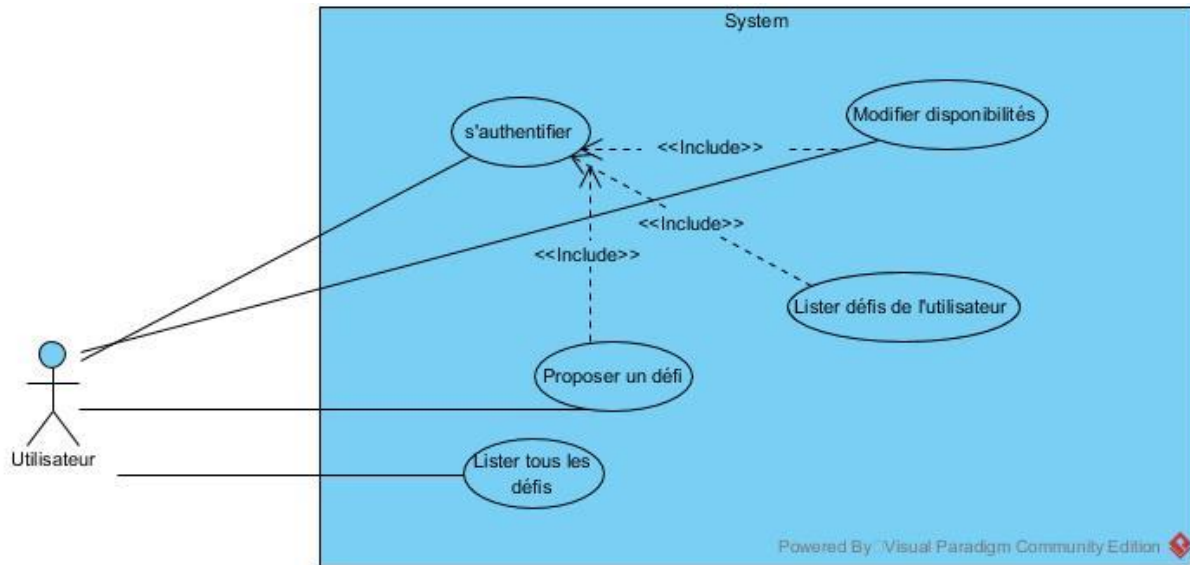
Afin d'assurer la persistance des données d'utilisation de l'application Web, une couche d'accès aux données doit être développée avec la présence d'un système de gestion de bases de données ainsi que les interfaces permettant son utilisation.

Pour ce dernier point, étant dans le cadre d'une maquette, il n'est pas prioritaire. Les données seront dans un premier temps stockées côté serveur applicatif et leur persistance ne sera pas assurée (perdues lorsque le serveur est coupé).

## Dossier de conception

### Cas d'utilisation

L'ensemble des fonctionnalités implémentées pour cette maquette peuvent être identifiées dans le diagramme de cas d'utilisation ci-dessous :



#### Description des cas d'utilisation :

- S'authentifier : permet à l'utilisateur de s'authentifier sur la plateforme
- Proposer un défi : permet à l'utilisateur authentifié de proposer un défi
- Lister tous les défis : permet à l'utilisateur authentifié ou non d'afficher en temps réel les défis créés
- Modifier disponibilités : permet à l'utilisateur authentifié de modifier ses disponibilités
- Lister défis de l'utilisateur : affiche à l'utilisateur authentifié la liste des défis auxquels il participe, cette liste correspond aux défis proposés pour sur les jours où il est disponible et où la météo est qualifiée de « bonne ».

## Architecture client-serveur

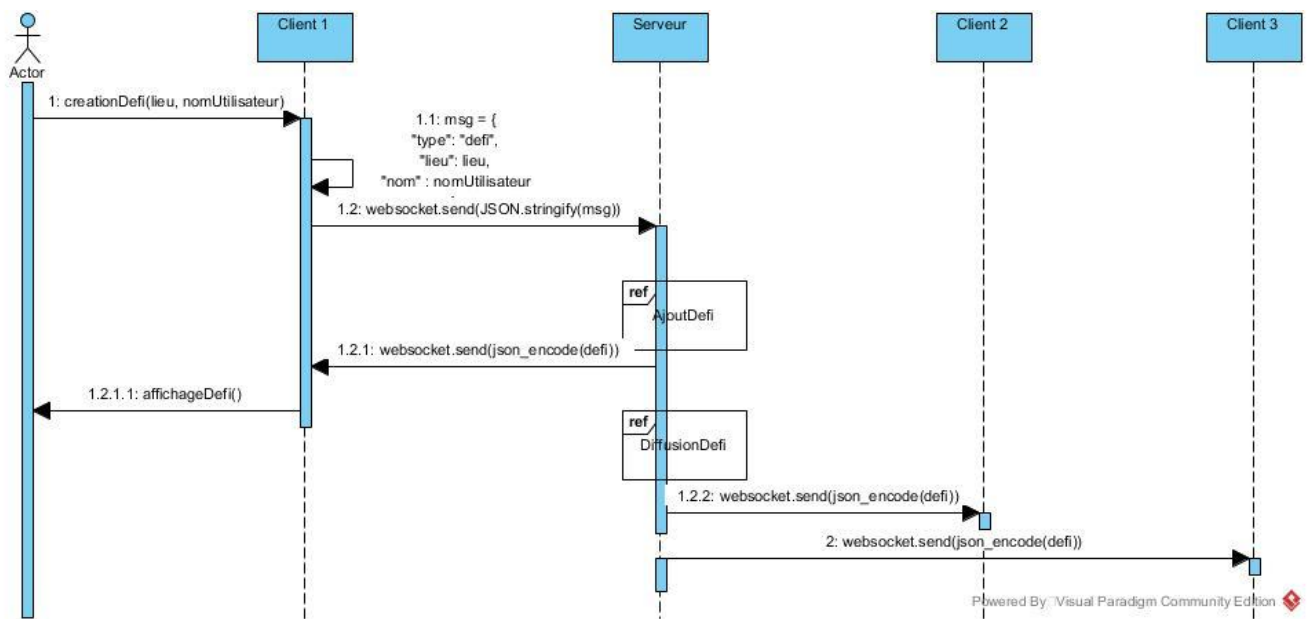
L'ensemble des fonctionnalités listées dans le diagramme précédent vont être implémentées grâce à des échanges passés sur le WebSocket, son utilisation nous permet de faire des appels de fonction côté serveur sur les clients en cas de modification initié par un des clients et ainsi permettre un affichage plus dynamique côté IHM en économisant des ressources (en évitant d'utiliser des techniques de polling par exemple).

Client et serveurs en écoute sur le port spécifié à la création du WebSocket des différents messages arrivants et des actions sont spécifiées à leurs arrivées.

Les messages échangés utilisent le format JSON afin de représenter l'information de manière structurée. Ceux-ci ont tous un champ « type » afin que le composant (client ou serveur) identifie le traitement à effectué lors de sa réception.

Le diagramme de séquence sur le cas proposer un défi illustre l'utilisation faite des websockets pour avertir tous les clients de la création d'un défi par un utilisateur :

Diagramme de séquence proposer un défi :



## Implémentation Front-End

L'ensemble des fonctionnalités et des échanges listées dans les diagrammes précédents doivent être affichés en temps réels. Cet affichage se fait par appel à des fonctions javascript utilisant la bibliothèque jQuery lors de la réception d'informations de la part du serveur ou de l'utilisateur via son IHM.

Aussi, l'application ne comporte donc qu'une seule page où les codes d'affichage html sont créés ou appelés par ces fonctions javascript.

### Partie IHM

La partie IHM du site a été développée avec l'aide du framework d'interface Bootstrap notamment pour les éléments qui contiennent du contenu, les formulaires ainsi que les boutons.

#### Charte graphique

Menus :

Utilisateur authentifié :




Mes disponibilités Proposer un défi Tous les défis

Utilisateur non authentifié :

S'authentifier

Jeu de couleurs :

Les couleurs suivantes sont utilisées pour le site :

Couleur	Code hexadécimal	Utilisation
	#ff5c5c	Eléments principaux
	#2ecc71	Eléments secondaires
	#484848	Police



Maquette de la page :

Utilisateur non authentifié :

S'authentifier

Lieu

Météo prévue

Lieu
















Météo prévue

Lieu

Météo prévue

↩ S'authentifier

## Liste des défis

Défi créé par: Ray					
Lieu: Lyon					
Participants					
Météo du 1554163200	Météo du 1554249600	Météo du 1554336000	Météo du 1554422400	Météo du 1554508800	
					
Défi créé par: Ray					
Lieu: Marseille					
Participants					
Météo du 1554163200	Météo du 1554249600	Météo du 1554336000	Météo du 1554422400	Météo du 1554508800	
					
Défi créé par: Tom					
Lieu: Strasbourg					
Participants					
Météo du 1554163200	Météo du 1554249600	Météo du 1554336000	Météo du 1554422400	Météo du 1554508800	
					

Active Windows  
Accédez aux paramètres pour activer Windows.

Utilisateur authentifié :

Jours de la semaine en vert si Disponible sinon en rouge

Zone disponibilités + affichage de mes défis

Zone pour proposer un nouveau défi

Zone pour rejoindre un défi

Mes disponibilités
Proposer un défi
Mes défis

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
-------	-------	----------	-------	----------	--------	----------

Lieu

Date

Participants

Lieu

Date

Participants

Lieu

Date

Participants

Proposer un défi

Lieu du défi

MAP

Liste des défis

Lieu	Participants	Météo prévue	Je participe
------	--------------	--------------	--------------

Lieu

Participants

Météo prévue

Je participe

Mes disponibilités Proposer un défi Tous les défis

## Mes disponibilités

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
-------	-------	----------	-------	----------	--------	----------

## Mes défis

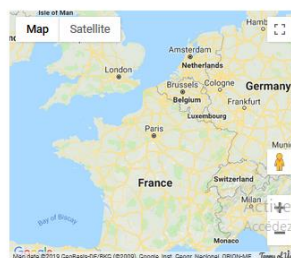
## Planifier un défi

Créer un défi école

Lieu du défi

Marseille, France

Quel défi



Activer Windows  
Accédez aux paramètres pour activer Windows.

Map data ©2019 OpenStreetMap contributors, Imagery ©2019 Google, Inst. Geogr. National, ORION-HE, Terms of Use

## Liste des défis

Défi créé par: Roy					
Lieu: Lyon					
Participants					
Météo du 15/04/16 13:20:00	Météo du 15/04/24 09:00	Météo du 15/04/30 00:00	Météo du 15/04/42 24:00	Météo du 15/04/08 08:00	

Défi créé par: Roy					
Lieu: Marseille					
Participants					
Météo du 15/04/16 13:20:00	Météo du 15/04/24 09:00	Météo du 15/04/30 00:00	Météo du 15/04/42 24:00	Météo du 15/04/08 08:00	

Défi créé par: Tom					
Lieu: Strasbourg					
Participants					
Météo du 15/04/16 13:20:00	Météo du 15/04/24 09:00	Météo du 15/04/30 00:00	Météo du 15/04/42 24:00	Météo du 15/04/08 08:00	

Activer Windows  
Accédez aux paramètres pour activer Windows.

Mentions légales

À propos

Infos pratiques

Interface avec le serveur :

Pour cette maquette, le client envoie au serveur les messages suivants :

Pour s'authentifier : fonction authentication()

*message échangé*

```
var msg = {  
    "type": "cclient",  
    "name": nomUtilisateur,  
    "password": mot de passe,  
};
```

Pour changer ses disponibilités : fonction changerDispo(jour)

*message échangé*

```
var msg = {  
    "type": "cdispo",  
    "name": nomUtilisateur  
    "jour" : jour,  
    "dispo" : 0 ou 1
```

Pour créer un défi : fonction creationDefi()

*message échangé*

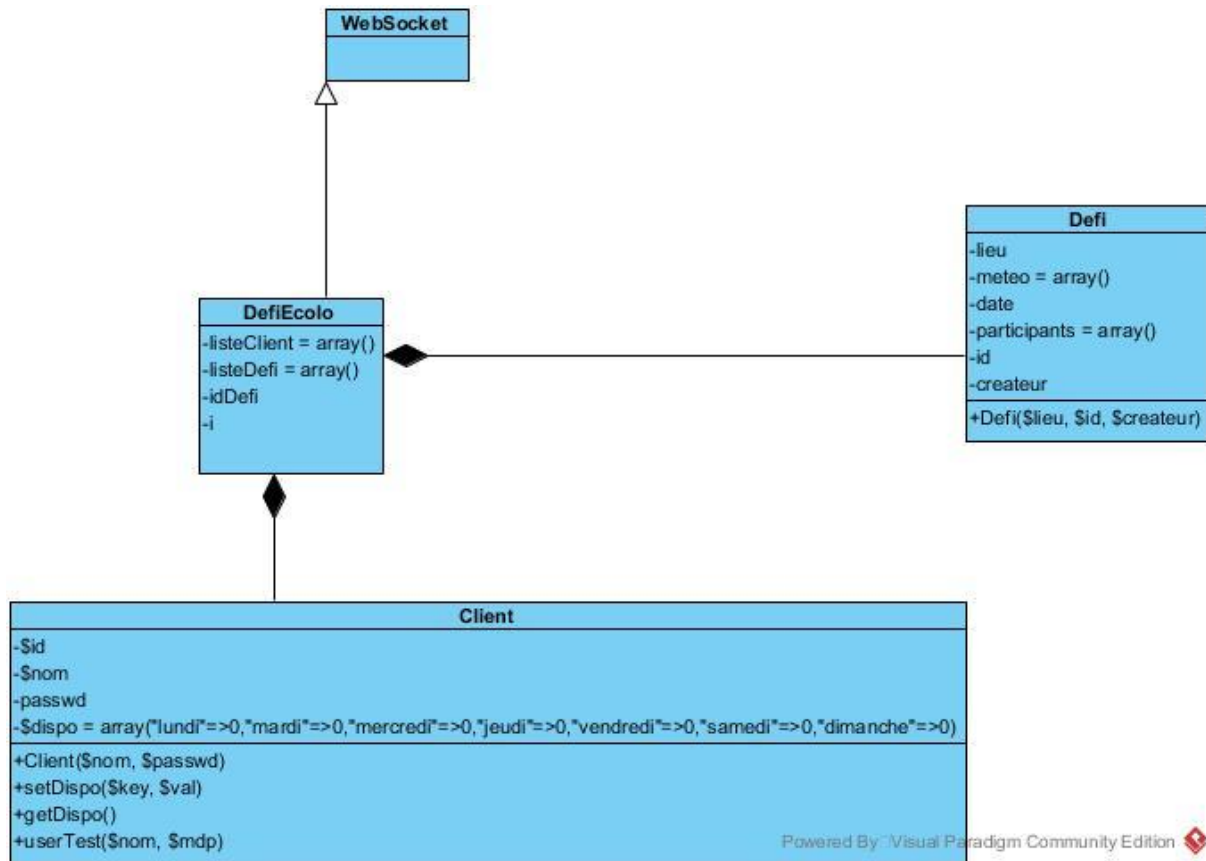
```
var msg = {  
    "type": "defi",  
    "lieu": lieu,  
    "nom" : nomUtilisateur  
};
```

## Implémentation Back-End

Côté serveur, c'est un serveur WebSocket codé en PHP qui a été utilisé, les fonctions json\_encode et json\_decode sont utilisés pour transmettre l'information aux clients ou pour transformer l'information reçue en JSON au format objet PHP.

Aussi, une architecture orientée objet a été utilisée.

Diagramme de classe PHP



## Interface avec les clients

Les messages sont envoyés sous forme d'array au client.

Réponse à une demande d'authentification :

*message échangé*

```
$phparr = array("type"=>"repAuth", "rep"=>$rep, "client"=>$this->listeClients[$nom]);
```

Diffusion d'un défi à tous les clients ou à un seul (selon la fonction appelée)

*message échangé*

```
$phparr = array("type"=>"creationDefi", "createur"=>$createur, "defi"=>$defi);
```

Ajout du défi aux défis d'un utilisateur

*message échangé*

```
$phparr = array("type"=>"ajoutDefi", "date"=>$jour, "meteo"=>$this->listeDefis[$idDefi]->meteo[$date]["icone"], "lieu"=>$this->listeDefis[$idDefi]->lieu);
```

## Interface OpenWeatherMap

Le serveur centralise les informations météo sur un défi une fonction est implémentée pour récupérer ces informations via l'API d'OWM, cette fonction retourne les informations dont on a besoin (c'est-à-dire les informations météo sur une heure donnée pour les 5 prochains jours) sous la forme d'un tableau :

```
array("J1"=>array("date"=>$json->list[3]->dt, "icone"=>$json->list[3]->weather[0]->icon), "J2"=>array("date"=>$json->list[11]->dt, "icone"=>$json->list[11]->weather[0]->icon), "J3"=>array("date"=>$json->list[19]->dt, "icone"=>$json->list[19]->weather[0]->icon), "J4"=>array("date"=>$json->list[27]->dt, "icone"=>$json->list[27]->weather[0]->icon), "J5"=>array("date"=>$json->list[35]->dt, "icone"=>$json->list[35]->weather[0]->icon));
```