



Ecole Polytechnique de l'Université de Tours  
64, Avenue Jean Portalis  
37200 TOURS, FRANCE  
Tél. +33(0)2 47 36 14 14  
Fax. +33(0)2 47 36 14 22  
[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr)

Rapport de projet de développement embarqué

# Arrosage Intelligent

Apprentis:  
Quentin Chalopin  
[Quentin.chalopin@etu.univ-tours.fr](mailto:Quentin.chalopin@etu.univ-tours.fr)  
Kévin Repillez  
[Kevin.repillez@etu.univ-tours.fr](mailto:Kevin.repillez@etu.univ-tours.fr)

Tuteur:  
Romain Ravaud  
[romain.ravaud@univ-tours.fr](mailto:romain.ravaud@univ-tours.fr)

## Sommaire

<b>Introduction</b>	<b>4</b>
<b>Contexte</b>	<b>4</b>
<b>L'origine du projet</b>	<b>4</b>
<b>Cahier des charges</b>	<b>5</b>
<i>Application graphique</i>	5
<i>Un arrosage programmable</i>	5
<i>Un Arrosage intelligent</i>	5
<i>Commandes manuelles</i>	6
<i>Multi plates-formes</i>	6
<i>Accessible depuis l'extérieur</i>	6
<b>Elaboration du projet</b>	<b>7</b>
<i>Choix de l'architecture</i>	7
<i>Une Web Application</i>	7
<i>Choix du serveur</i>	7
<i>Choix technologiques et des langages de programmation</i>	8
Html, Php, JQuery et CSS	8
Jquery Mobile	9
AJAX	9
C++ et librairies « Cooking Hacks »	9
Base de données	9
Crontab	11
Protocole de communication	11
<i>Elaboration des IHM de l'application</i>	12
Page d'accueil	12
Page de commandes manuelles	13
Page de configuration des vannes	14
Page statistique	15
Page de programmation	16
Enchaînement des IHM	17
<b>Résultats et tests</b>	<b>18</b>
<i>Programmation et enchaînement des actions</i>	18
La page d'accueil	18
La page statistique	19
La page de commande manuelle	20
La page de création d'interrupteurs	22
La page de suppression d'interrupteurs	24
La page programmation	26
<i>Schéma de la base de données</i>	29
<b>Les axes d'améliorations</b>	<b>30</b>
<i>La gestion des commandes manuelles :</i>	30
<i>La visualisation de la programmation :</i>	30
<i>Le nombre d'arrosage :</i>	30
<i>Gestion du changement d'heure :</i>	30
<b>Annexes</b>	<b>30</b>



## Introduction

Dans le cadre de notre formation par apprentissage à Polytech'Tours en Informatique Industrielle, nous devons réaliser un projet orienté « Développement embarqué ». Nous avons la possibilité de proposer un projet personnel ou en partenariat avec notre entreprise.

Nous avons donc pensé un projet dans le domaine de la domotique permettant la gestion de la programmation d'un arrosage automatique qui se dit « intelligent ». Ce projet s'inscrit sur deux tableaux, celui du projet « développement » mais aussi sur le projet « électronique ». Tous deux seront réalisés dans le cadre de la formation de Polytech'Tours.

## Contexte

D'une manière générale, la domotique est un sujet très à la mode de nos jours. De plus en plus de projets/produits sont commercialisés pour le grand public dans cette catégorie. La domotique peut regrouper énormément de domaines différents. Dans beaucoup de cas, la domotique est utilisée dans les foyers pour commander des lumières, volets ou tout autre objet qui nous entoure de près ou de loin. Par exemple, il serait envisageable de commander des interrupteurs permettant l'allumage/extinction de l'arrosage de la pelouse de sa maison. Et mieux encore, pouvoir programmer à l'avance la gestion de cet arrosage. En plus d'être automatisé mécaniquement et/ou électriquement, il serait possible de rajouter de « l'intelligence » au système afin d'obtenir un arrosage précis, optimisé et intelligent.

## L'origine du projet

Le projet de réaliser un arrosage intelligent et autonome est issu d'un besoin personnel. En effet, nous avons tous deux un système d'arrosage intégré à nos jardins avec une commande d'ouverture des vannes manuelles ou avec un petit programmeur. Nous nous sommes dit qu'il serait intéressant d'automatiser ces ouvertures via une application utilisable sur un ordinateur ou sur un Smartphone. De plus, nous voulions que cette application permette la gestion de l'arrosage en fonction des conditions météorologiques. En effet, nous avons remarqué que la plupart des programmeurs ne prennent pas en compte l'environnement extérieur et ne sont pas programmable à distance.

## Cahier des charges

Dans ce chapitre, nous allons expliquer les principaux éléments qui composeront notre futur système.

### Application graphique

Premièrement, notre système « intelligent » sera composé d'une application graphique. Cette application est la seule partie pouvant communiquer avec l'utilisateur. En effet, elle permet de faire le lien avec tout autre élément composant le système d'arrosage automatique.

Le côté graphique permet également une meilleure compréhension pour l'utilisateur et il pourra donc configurer plus aisément son système.

### Un arrosage programmable

C'est ici la fonctionnalité la plus importante du système. Notre système donnera la possibilité à l'utilisateur de « programmer » son arrosage. C'est-à-dire qu'il pourra configurer le système pour qu'il allume (et éteigne) l'arrosage suivant une heure de début et de fin. Bien sûr, l'utilisateur pourra également choisir une ou plusieurs journées dans la semaine.

Lorsque l'utilisateur configure une journée d'arrosage, celle-ci sera répétée chaque semaine. Par exemple, si il décide de programmer l'arrosage le lundi de 12:00 à 12:20, la configuration sera active pour chaque lundi de l'année.

Pour retirer ou modifier une configuration, l'utilisateur aura accès à une interface permettant ces actions.

### Un Arrosage intelligent

En plus d'être programmable, le système doit être « intelligent ». On peut imaginer que l'arrosage sera dynamique en fonction des conditions météorologiques et notamment en fonction de l'humidité de la pelouse. L'utilisateur pourra choisir un taux d'humidité limite pour lequel le système arrêtera l'arrosage automatiquement.

Suivant le temps restant on peut imaginer que d'autres valeurs pourront être prise en compte comme la température ou encore la luminosité.

## Commandes manuelles

L'application devra également offrir la possibilité à l'utilisateur d'allumer et fermer l'arrosage du système à tout moment. Ce cas d'utilisation mettra à jour en direct l'ouverture et fermeture des vannes (ou interrupteurs).

## Multi plates-formes

L'application devra être accessible depuis toutes les plates-formes informatiques. Notamment depuis un ordinateur mais aussi depuis un Smartphone Apple ou Android. Selon le support, l'application devra s'adapter pour que le contenu reste lisible.

## Accessible depuis l'extérieur

Nous devons donc réaliser une application utilisable sur toutes les plates-formes existantes, avec la possibilité de gérer cette application à distance.

Notre application devra avoir la possibilité de commander toutes les vannes manuellement, ainsi que de visualiser l'état actuel des vannes.

## Elaboration du projet

### Choix de l'architecture

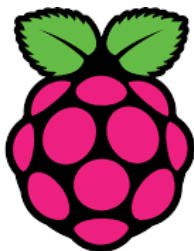
Nous avons choisi de réaliser cette application en séparant les différentes tâches. Ce projet se sépare donc en trois modules. Un module garage permettant de recevoir les informations d'ouvertures des vannes via un module de communication sans fil. Un module météo totalement autonome permettant de transmettre les informations liées à l'environnement du jardin, ces informations seront transmises via un module de communication identique à celui du module garage. Ces deux modules seront réalisés lors du projet électronique. Enfin, un module maison synchronisant les informations reçues par les différents modules, et gérant l'interface avec l'utilisateur.

### Une Web Application

Nous devons réaliser une application accessible depuis n'importe quels appareils numériques ayant une connexion internet. Cela concerne donc tous les Smartphones, tablettes et ordinateurs. Tous ces appareils fonctionnent sous différents systèmes d'exploitation. Nous avons donc choisi de réaliser cette application sur une page web, pour différentes raisons. Premièrement, une application web nous semblait plus simple à réaliser, car il existe de nombreuses applications similaires qui pouvaient nous aider dans cette réalisation. Ensuite, nous souhaitons une application utilisable par tous et à tous moments sans nécessairement devoir installer quelconques programmes pour pouvoir l'utiliser.

Ce choix a donc impliqué que le module maison soit relié à internet, et la mise en place d'un serveur hébergeant l'application.

### Choix du serveur



Nous avons choisi un Raspberry Pi, car ce projet est d'origine personnelle et que nous avons déjà un tel équipement correspondant parfaitement à nos besoins. Nous avons choisi d'installer la solution « Raspbian » comme système d'exploitation. Celui-ci est allégé pour le Raspberry et totalement customisé pour son architecture.

## RaspberryPi

Pour pouvoir accéder au serveur depuis l'extérieur, nous utilisons une BBox (Bouygues Télécom) qui offre une adresse IP Statique par abonnement. Nous avons utilisé la méthode de **redirection de port** pour que l'utilisateur connaissant la web application puisse y accéder depuis l'extérieur.

Nous avons redirigé les ports suivants :

- 80 ➔ Protocol http : Pour que l'utilisateur puisse accéder à l'application depuis l'extérieur (requête http)
- 22 ➔ Secure Shell : Très utile pour accéder au serveur sur un terminal depuis l'extérieur

## Choix technologiques et des langages de programmation

### *Html, Php, JQuery et CSS*

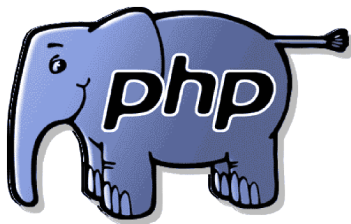
Ces 4 langages sont quasi incontournables pour la construction d'une page web dynamique et interactive :

- HTML et CSS



La justification est toute faite, ces deux langages sont indispensables pour construire un site web.

- PHP :



Ce langage interprété du côté serveur est très utile pour exécuter beaucoup d'actions, notamment avec la base de données. Ce langage est un formidable outil pour développer une Web Application agréable à utiliser.

- JQuery



Créé à partir de JavaScript, ces bibliothèques permettent de réaliser des actions plus simplement qu'avec JavaScript. Il est à noter également que les méthodes sont plus explicites et offrent parfois, une meilleure compatibilité avec les navigateurs web.



### *Jquery Mobile*



jQuery Mobile est un framework créé pour adapter facilement et efficacement une application web pour notre Smartphone. Grâce à celui-ci, l'utilisateur va pouvoir utiliser sa page web comme une application à part entière.

### *AJAX*

AJAX ou encore Asynchronous JavaScript And XML est une technologie ayant quelques avantages très utiles. Notamment celui de pouvoir mettre à jour du contenu HTML sans recharger la page. En plus de cela, nous pouvons choisir si les appels sont asynchrones ou synchrones. Lorsqu'un appel est asynchrone, l'application peut continuer à fonctionner même si le résultat n'est pas encore connu. En revanche un appel synchrone va « bloquer » le programme tant que le processus n'est pas terminé.

### *C++ et librairies « Cooking Hacks »*



Cooking Hacks est une entreprise fabriquant des composants et des cartes électroniques notamment des « shield » pour Arduino et Raspberry Pi. Nous avons utilisé l'un de ces shield Raspberry pour pouvoir brancher notre module xBee.

Cooking Hacks propose également ses propres librairies C++ adaptées à leurs shield. Cela nous permettra de développer facilement l'envoi et la réception des données via module xBee.

### *Base de données*

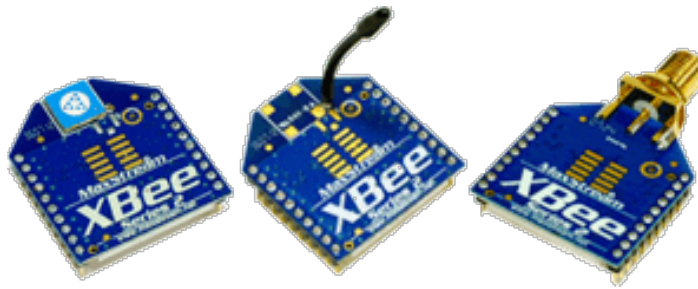


Pour la base de données, nous avons choisi d'utiliser MySQL. Cette solution s'installe facilement sur Raspberry Pi.

En plus de MySQL, nous avons installé PhpMyAdmin qui nous permet de gérer notre base de données facilement et rapidement. Cette interface graphique est beaucoup plus agréable que par terminal depuis le serveur.



### Module de communication sans fil



Il existe de nombreux moyens de communication sans fil tel que le WIFI, l'infrarouge, le Bluetooth, etc. Nos besoins pour ce module sont :

- une portée assez importante
- un signal qui ne peut être bloqué par un mur par exemple (certains modules se trouvent à l'intérieur et d'autres à l'extérieur)
- la vitesse de transfert de données n'est pas très importante
- un prix faible
- une mise en place simple
- la possibilité d'installer des modules relais pour étendre la portée du signal

	PORTEE	Visibilité direct	Vitesse	Configuration	Prix	Relais
<b>WIFI</b>	91 à 152 mètres selon la norme	NON	Elevée	Complexe	Elevée	NON
<b>INFRAROUGE</b>	Diminution des performances selon la distance	Les émetteurs et récepteurs doivent être presque alignés	Faible	Simple	Faible	NON
<b>BLUETOOTH</b>	Meilleure performance avec la distance que l'infrarouge	Fonctionnement possible sans visibilité directe	Elevée, mais moins que le WIFI	Simple	Faible	NON
<b>XBEE</b>	Environ 150 mètres	NON		Simple	Faible et en stock à polytech	OUI

Nous avons choisi les modules de communication XBEE. En effet, notre système nécessite un moyen de communication sans visibilité directe, avec une portée relative importante (ou la possibilité de relayer l'information), un prix faible et une configuration la plus simple possible. La vitesse de transfert de données n'est pas une contrainte importante pour ce projet.

### *Crontab*

Crontab est le diminutif de chrono table qui signifie table de planification. Cet outil est souvent utilisé sous Linux pour planifier différentes tâches à exécuter à un certain moment.

Dans notre projet, il nous permettra d'exécuter un script php qui gère l'ouverture et la fermeture des vannes. Nous avons choisi cette solution afin d'éviter de surcharger le temps processeur de notre serveur avec une tâche de fond. De ce fait, on configurera le scheduler pour exécuter ce script php toutes les minutes et ainsi déclencher (ou non) l'ouverture et la fermeture des vannes programmées.

### *Protocole de communication*

En plus de nos modules xBee, nous avons décidé de se fixer un protocole de communication commun à tous nos futurs modules. Le but est de partir sur des messages simples et facilement compréhensibles.

Nous avons choisi le pattern suivant :

**IdModule\_IdBroche\_Valeur**

Ainsi, le module cible comprendra si le message lui est destiné ou non (un id module lui sera fixé). Si cela correspond à son identifiant, le module changera l'état de la broche.

## Elaboration des IHM de l'application

Avant de commencer à programmer nos différentes pages nous avons voulu créer via un logiciel adapté (Cacoo) une première ébauche de notre application. Cela nous a permis d'avoir une base de travail commune.

### *Page d'accueil*

Composée de trois onglets, commandes manuelles, programmation et statistique renvoyant sur les pages associées. Cette page devra aussi gérer l'authentification des utilisateurs. De plus, comme sur toutes les pages nous insérerons un bandeau en en-tête avec un logo.



Figure 1 Page d'accueil

### Page de commandes manuelles

Elle permettra de commander manuellement toutes les vannes programmées par l'utilisateur et uniquement celles-ci. Cela permettra d'éviter d'avoir une page surchargée par une quantité de vannes inutiles. Etant donné que cette page n'affiche que les vannes actives, elle devra aussi permettre à l'utilisateur d'en ajouter ou d'en supprimer via un lien vers une autre page. Cette page permettra également de visualiser l'état actuel des vannes.



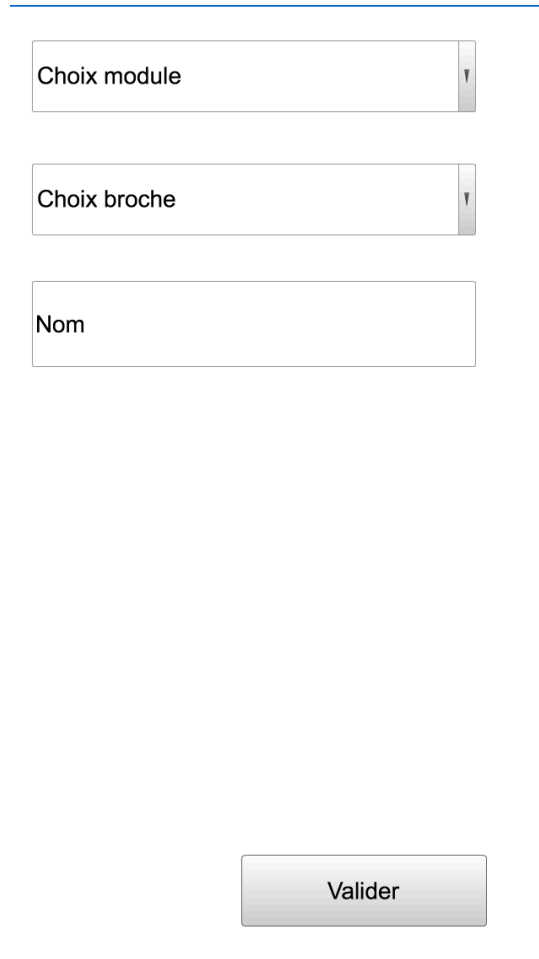
The screenshot displays a web interface for manual valve control. At the top, there are two valve controls: 'Vanne 1' with a blue 'ON' toggle switch, and 'Vanne 2' with a grey 'OFF' toggle switch. Below these, there is a dark blue sidebar menu titled 'VANNE' containing three buttons: 'Edit', 'Modif', and 'Create'. At the bottom of the interface is a large, light grey button labeled 'Retour accueil'.

Figure 2 Page de commandes manuelles

### *Page de configuration des vannes*

Dans un premier temps, elle permettra à l'utilisateur de choisir un module. En effet, nous avons imaginé la possibilité d'avoir plusieurs modules garage ou la possibilité de commander des contacteurs, comme l'allumage d'une lumière par exemple via les modules météo.

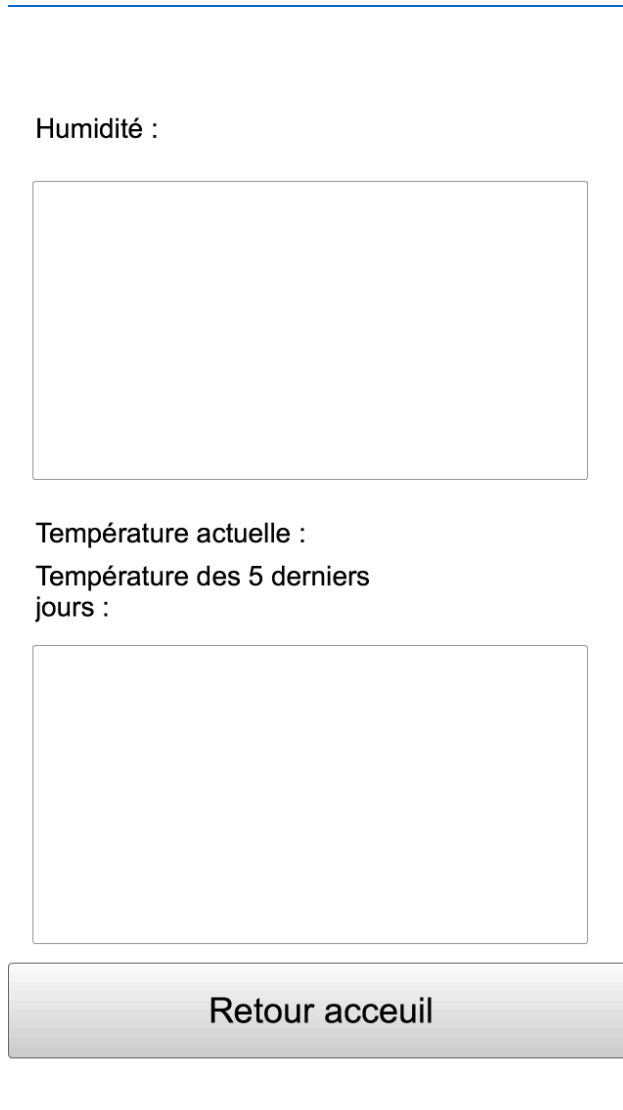
Ensuite, l'utilisateur pourra choisir la broche sur laquelle se trouve la vanne. Pour terminer nous souhaitons donner la possibilité à l'utilisateur de nommer les vannes, car nous pensons que cela simplifie l'utilisation.

The image shows a web form for configuring valves. It is enclosed in a thin blue border. Inside, there are three input fields stacked vertically. The first two are dropdown menus with a small downward arrow icon on the right; the first is labeled 'Choix module' and the second 'Choix broche'. The third is a standard text input field labeled 'Nom'. At the bottom right of the form area is a grey button with the text 'Valider'.

**Figure 3** Page de configuration des vannes

### *Page statistique*

Cette page permettra à l'utilisateur de suivre l'évolution des conditions météorologiques, ce suivi se ferait sur 24h avec un relevé toutes les heures. Cette page étant un plus pour l'application.



Humidité :

Température actuelle :  
Température des 5 derniers  
jours :

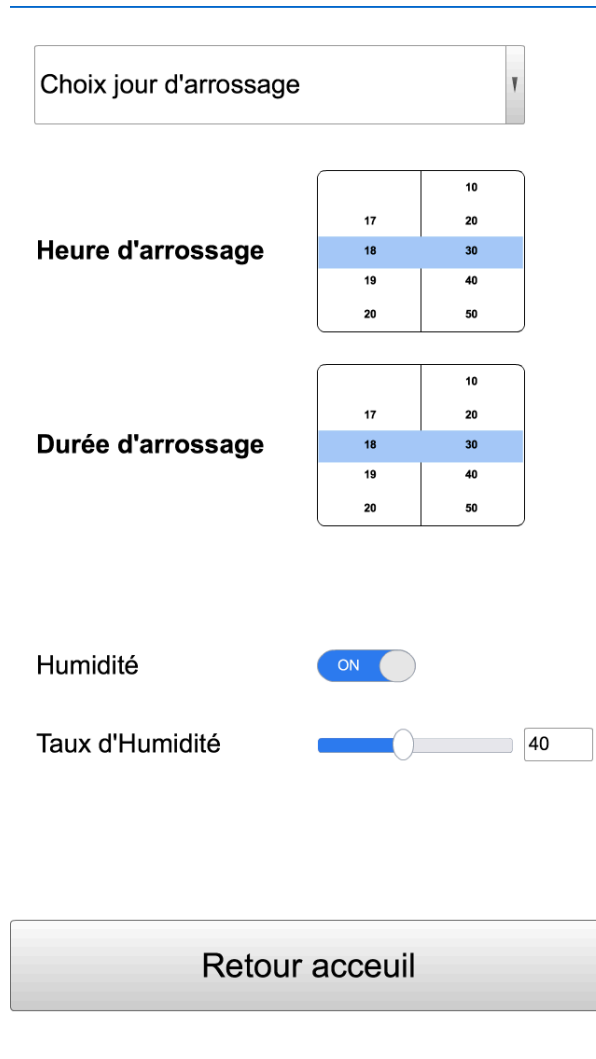
Retour accueil

The mockup shows a vertical container with a thin blue top border and a thin black bottom border. Inside, the text 'Humidité :' is followed by a large empty rectangular box. Below this, the text 'Température actuelle :' and 'Température des 5 derniers jours :' is followed by another large empty rectangular box. At the bottom of the container is a grey button with the text 'Retour accueil'.

Figure 4 Page de statistiques

### Page de programmation

La première partie de cette page est dédiée aux choix des jours, de l'heure et de la durée de l'arrosage. Cela permet de créer un calendrier d'arrosage hebdomadaire, ce calendrier sera répété toutes les semaines jusqu'à un changement de programmation. La deuxième partie sera le choix de l'arrosage par taux d'humidité (dans un premier temps) par l'intermédiaire d'un Switch et d'un Slider pour choisir le taux d'humidité voulu.



The screenshot shows a web interface for programming an irrigation system. It includes a dropdown menu for selecting the day of the week, two 5x2 grids for selecting the hour and duration of irrigation, a toggle switch for humidity control, a slider for setting the humidity rate, and a 'Retour accueil' button at the bottom.

Choix jour d'arrosage

Heure d'arrosage

17	10
18	20
19	30
20	40
	50

Durée d'arrosage

17	10
18	20
19	30
20	40
	50

Humidité

ON

Taux d'Humidité

40

Retour accueil

Figure 5 Page de programmation



## Enchaînement des IHM

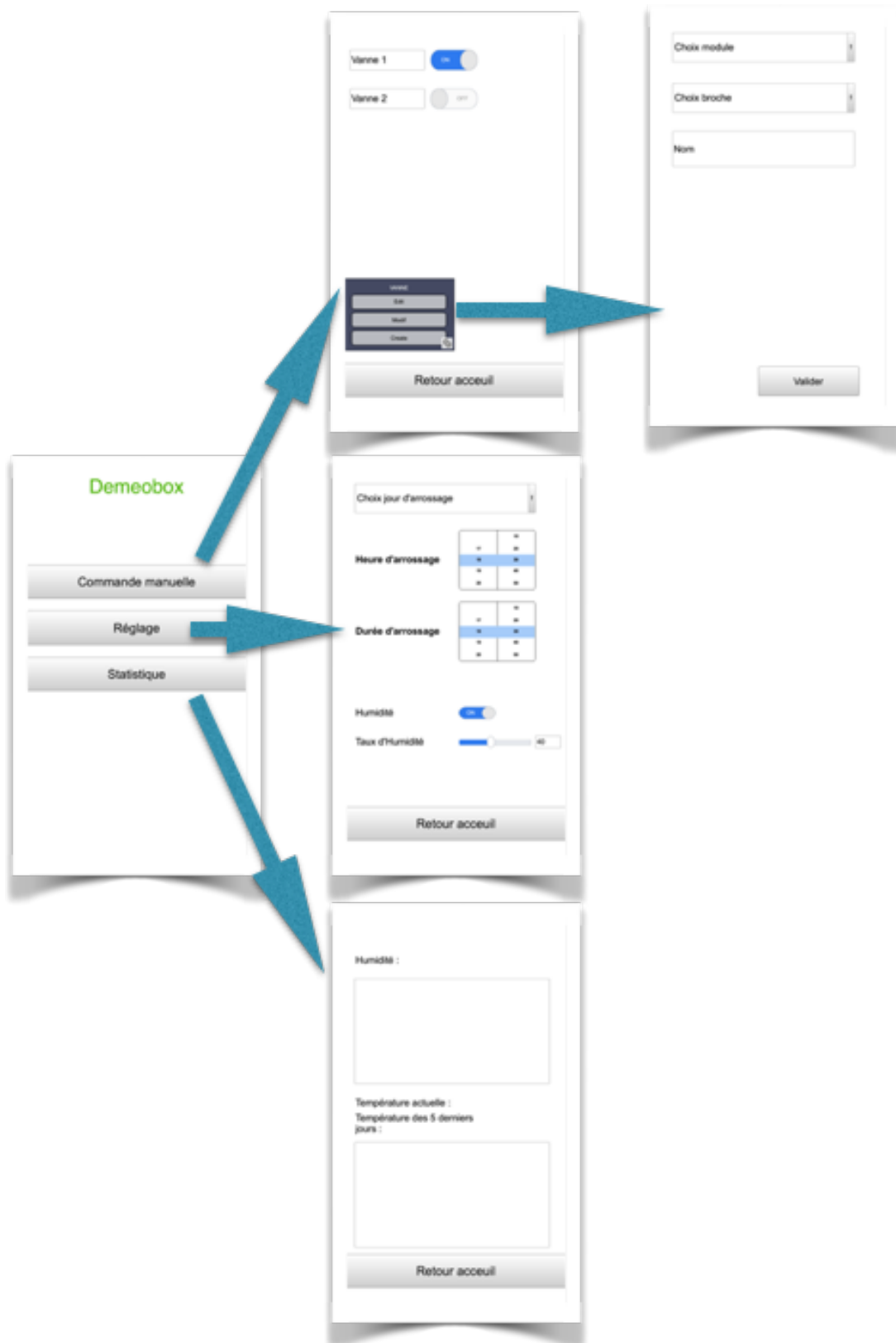


Figure 6 Diagramme d'enchaînement des pages

## Résultats et tests

### Programmation et enchaînement des actions

#### *La page d'accueil*

Elle est composée des trois onglets prévus. C'est une page HTML simple où chaque onglet est un lien vers une autre page HTML.

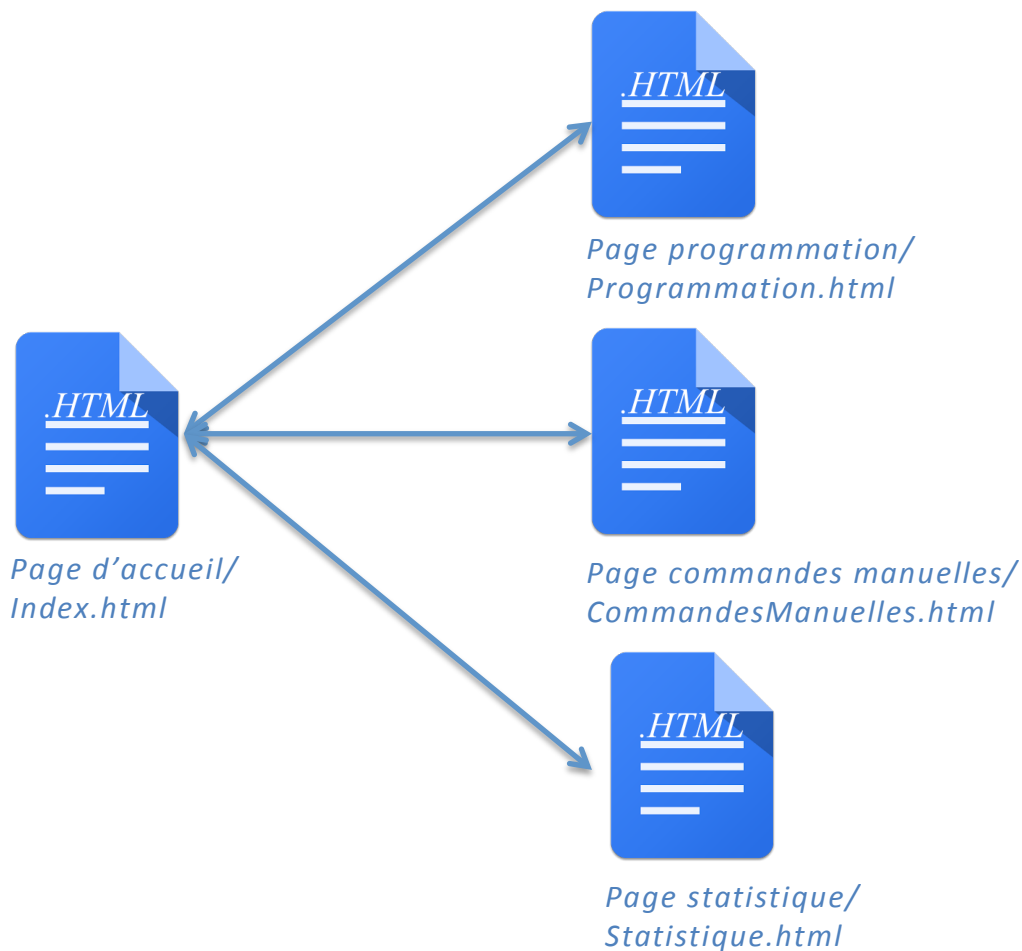


Figure 7 Schéma de liaison avec index.html

### La page statistique

La page HTML a été créée et elle fonctionne, en revanche la réalisation n'est à l'heure actuelle pas terminée. Nous souhaitons, pour réaliser ces graphiques, utiliser une page PHP qui récupère les données via une requête SQL puis génère une image. Cette image, sera affichée sur la page HTML. Les données de ce graphique étant récupérées par une requête SQL, il faudra s'assurer que les valeurs dans la base de données correspondent bien aux 24 derniers relevés d'informations.

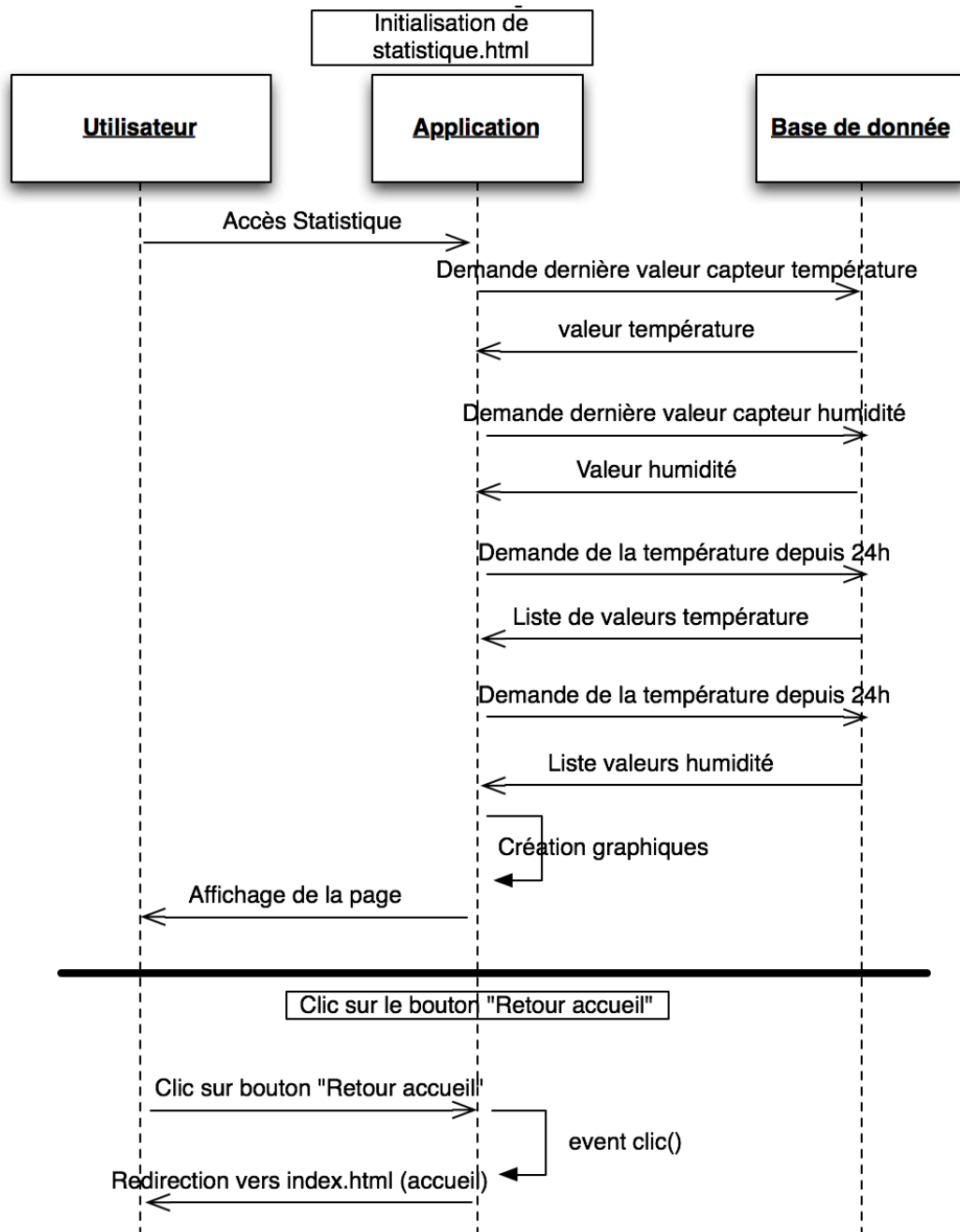


Figure 8 Diagramme de séquence de statistique.html

### La page de commande manuelle

Cette page fonctionne comme prévu dès le début du projet. Effectivement, elle permet de visualiser l'état des vannes enregistrées par l'utilisateur. L'état de ces vannes est représenté par un switch On/Off. Elles sont identifiées par un nom donné par l'utilisateur.

Elle permet à l'utilisateur d'accéder aux pages de création et de suppression d'interrupteurs.

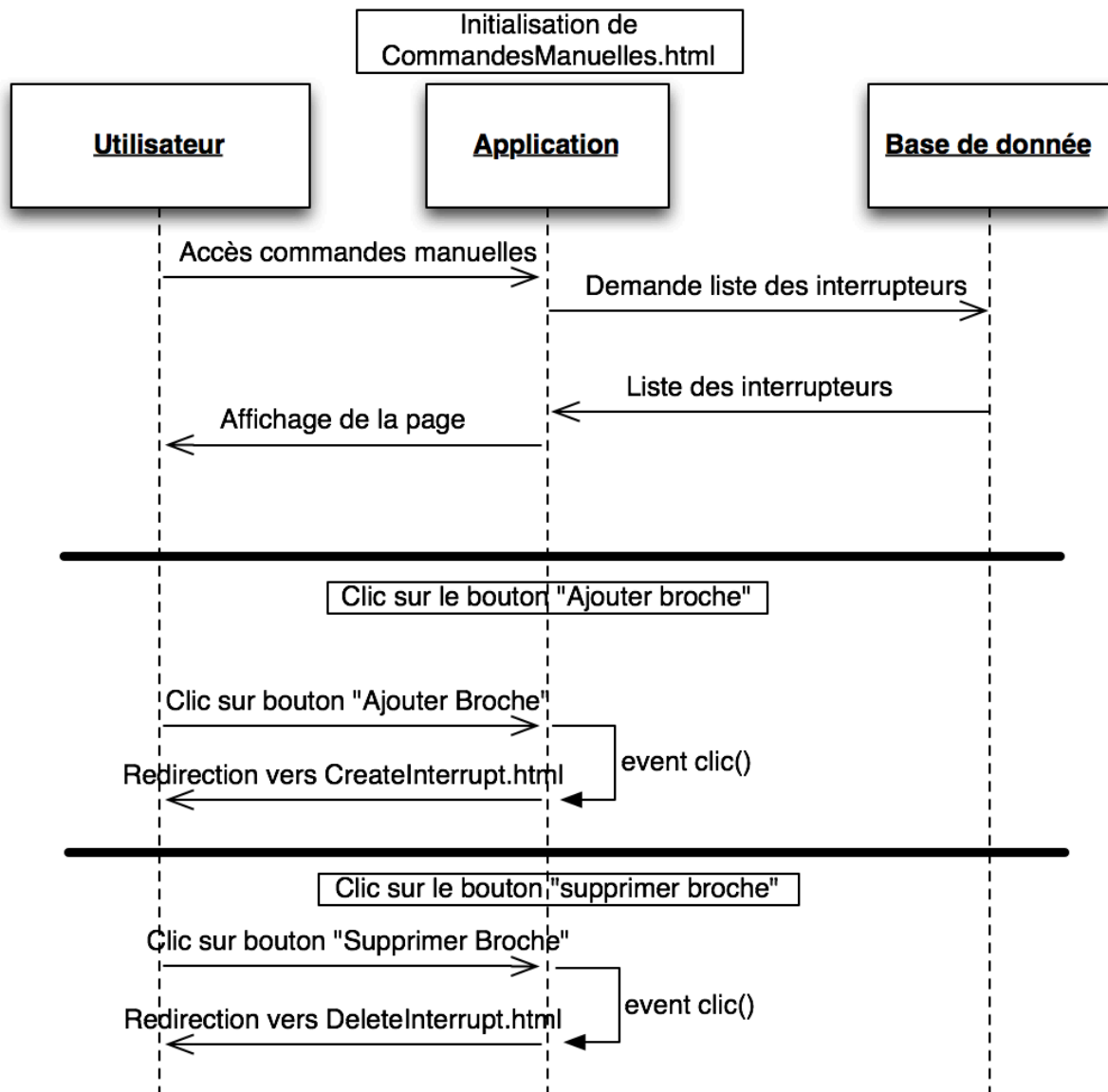


Figure 9 Diagramme de séquence de **CommandesManuelles.html**

Changement de valeur d'un interrupteur

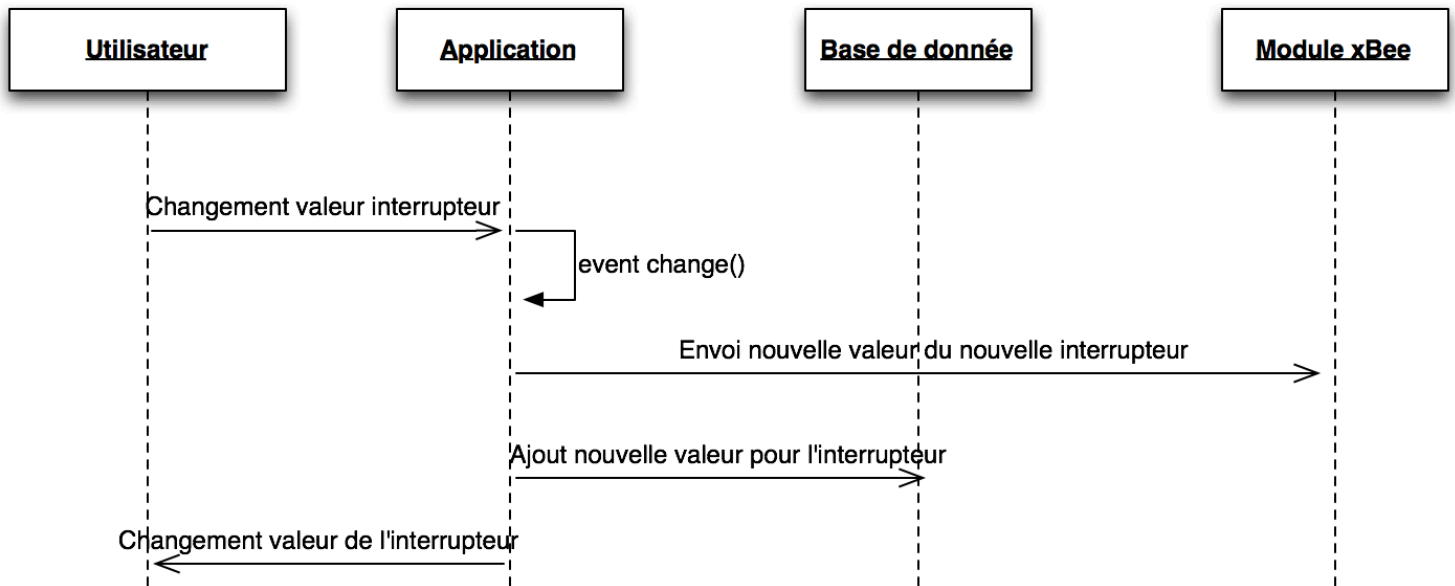


Figure 11 Diagramme de séquence de commandesManuelles.html

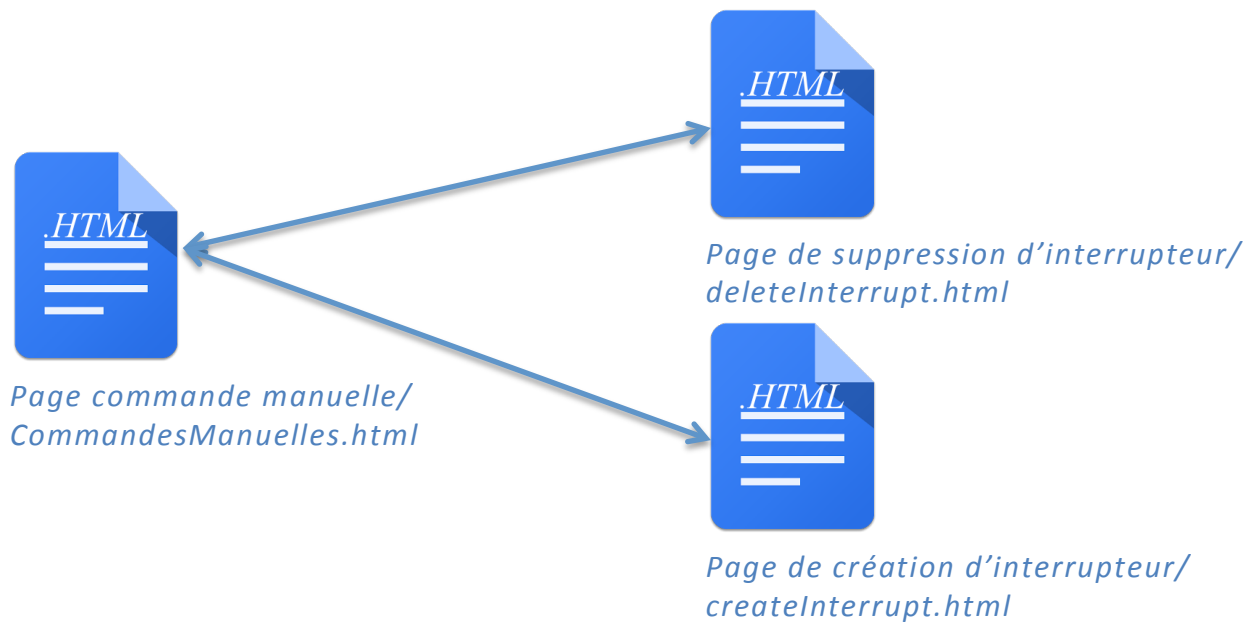
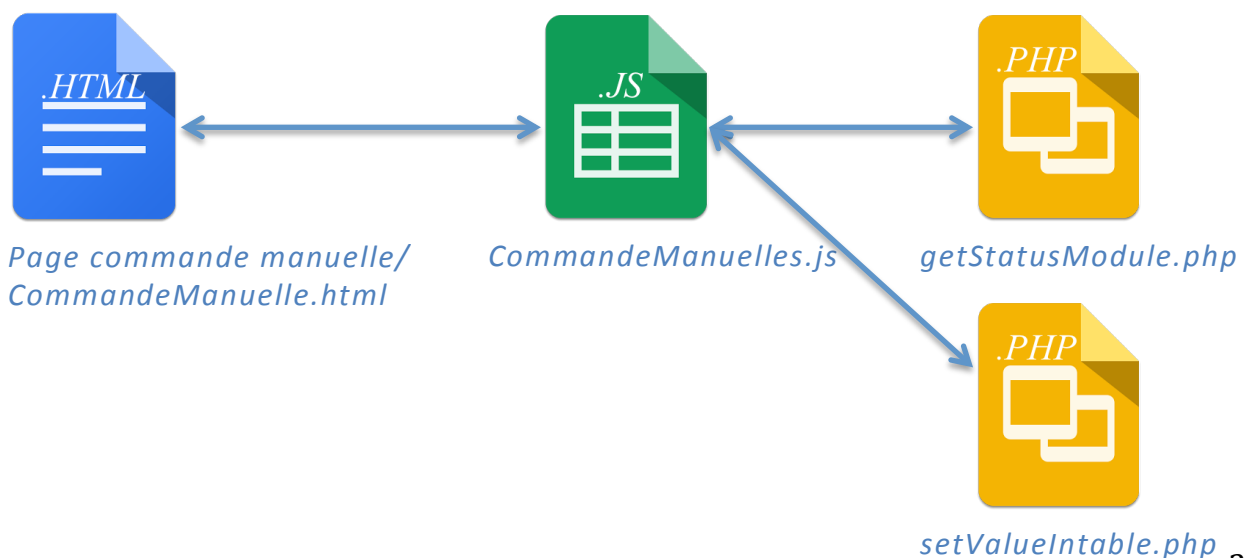


Figure 12 Schéma de liaison avec CommandesManuelles.html



### La page de création d'interrupteurs

Sur cette page, l'opérateur a la possibilité de configurer une nouvelle vanne. Pour cela il doit tout d'abord, choisir sur quel module se trouve la vanne souhaitée. Ce choix se fait via un menu déroulant dynamique, car il n'affiche que les modules enregistrés.

Le principe de menu déroulant dynamique est repris pour l'affichage des vannes correspondant au module choisi. Ce menu ne présente que les vannes encore programmables.

Il doit également choisir un module météo à associer à cette vanne, cela permet la création de différents secteurs. Cette option permet d'ajouter une fonctionnalité au projet. Grâce à cette option, l'utilisateur va pouvoir créer des secteurs et donc affiner ses réglages.

Enfin, l'utilisateur choisit un nom significatif pour la vanne.

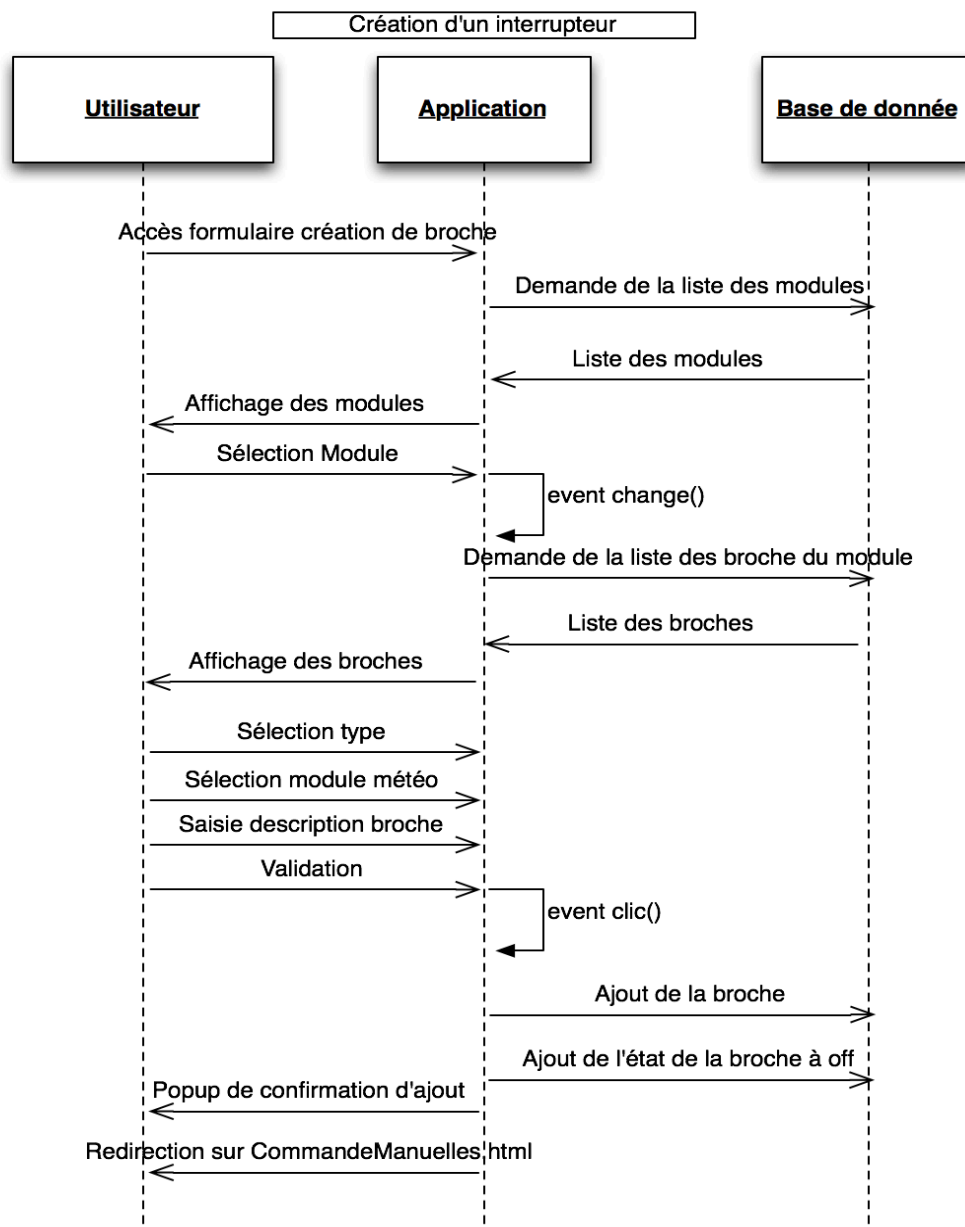


Figure 13 Diagramme de séquence de création d'un interrupteur

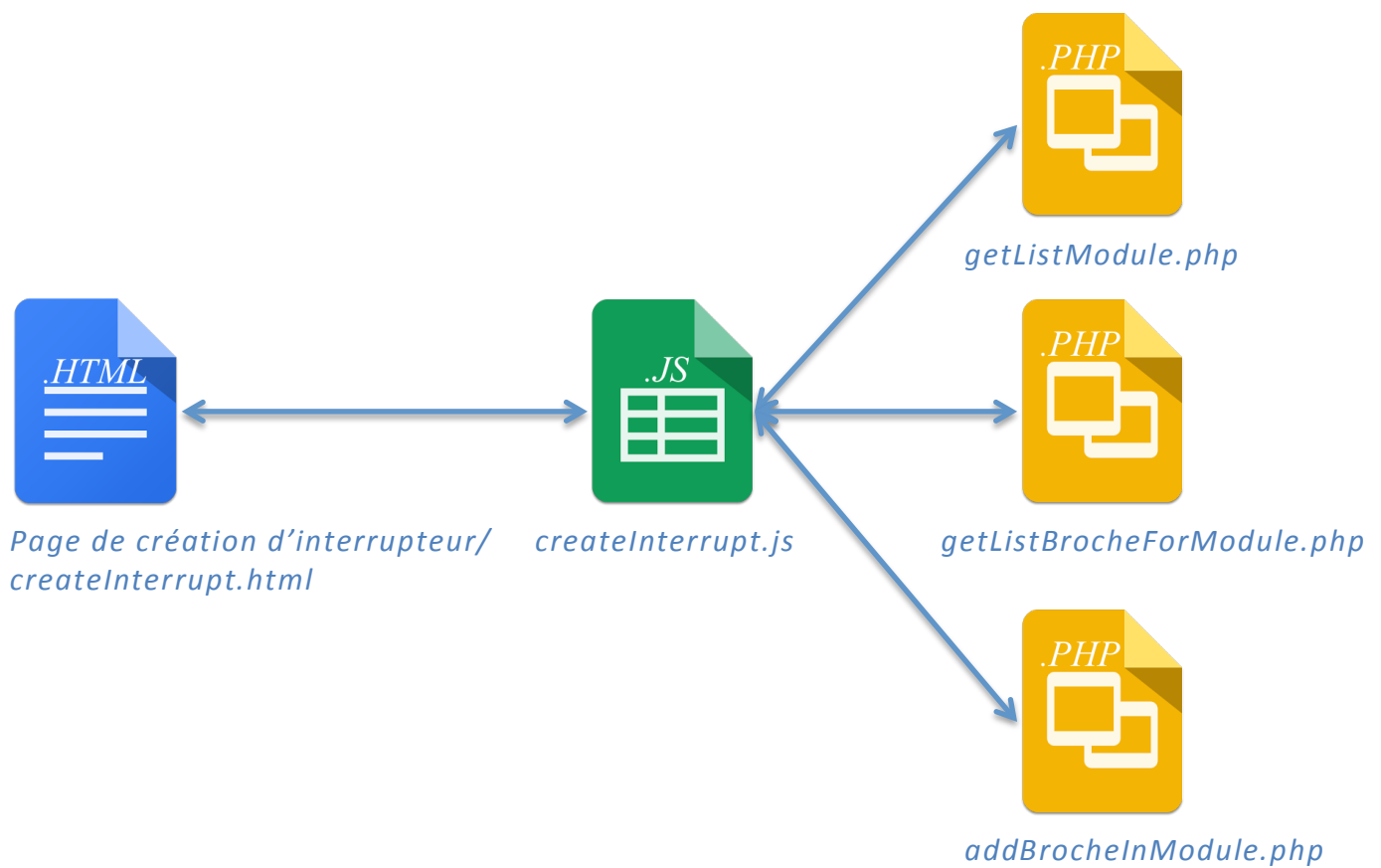


Figure 14 Schéma de liaison avec createInterrupt.html

### La page de suppression d'interrupteurs

Cette page permet de supprimer une vanne. Il faut donc d'abord choisir sur quel module elle se trouve. Une fois ce choix fait, la liste des broches programmées sur ce module est affichée, il ne reste plus qu'à sélectionner la vanne souhaitée et la supprimer.

La base de données est ensuite mise à jour, permettant de visualiser à nouveau la vanne supprimée comme vanne programmable.

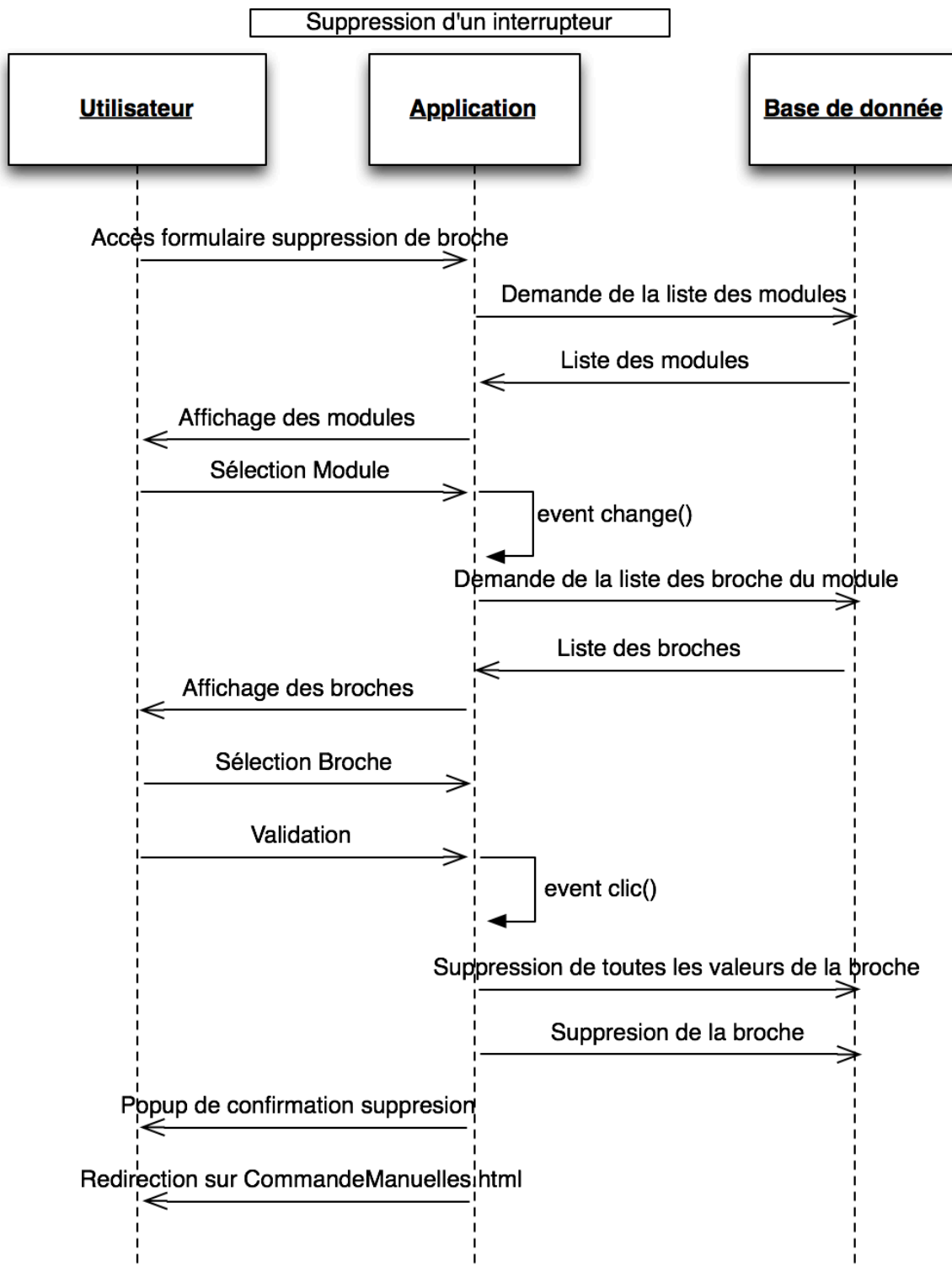


Figure 15 Diagramme de séquence de suppression d'un interrupteur



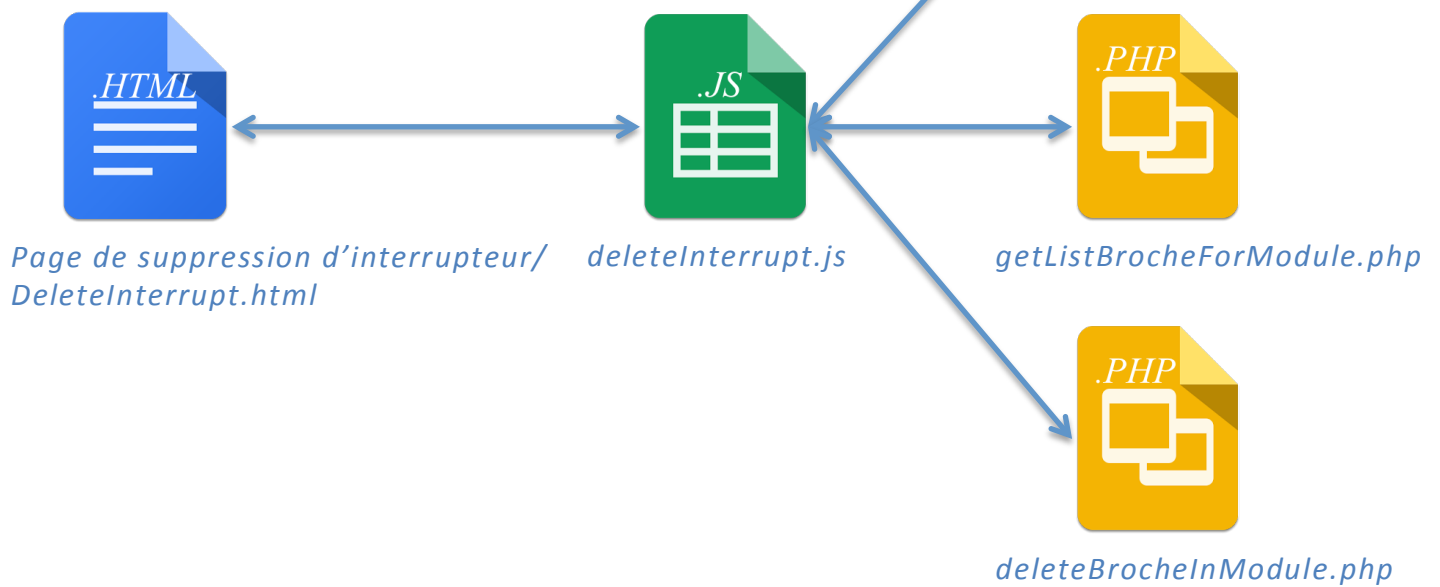


Figure 16 Schéma de liaison avec deleteInterrupt.html

### *La page programmation*

Cette page est le cœur de l'application. Comme pour les pages précédentes, on retrouve tout d'abord un menu déroulant permettant de choisir sur quel module nous souhaitons créer un programme d'arrosage. Ce choix de module permet à l'utilisateur d'avoir plusieurs modules garage, ou de programmer des interrupteurs sur les modules météo.

Le second menu permet de choisir quelle vanne est à programmer. Par défaut, toutes les vannes activées sont sélectionnées. Le fait de pouvoir choisir les vannes à ouvrir permet de créer une sectorisation de l'arrosage via la programmation. Par exemple on peut choisir d'arroser une partie du jardin le lundi et le reste le mercredi.

Pour le choix du jour d'arrosage, l'utilisateur peut choisir les jours d'arrosage qu'il souhaite.

En revanche, il n'est pas possible de créer un planning d'arrosage où l'on ouvre deux fois la même vanne dans la même journée.

L'utilisateur a aussi la possibilité de choisir d'arroser via les capteurs d'humidités. Lorsqu'il choisit cette option, une fenêtre popup s'ouvre. Elle informe que la durée d'arrosage ne sera pas prise en compte dans cette configuration. Cela implique aussi que les vannes ne se fermeront pas toutes au même moment car il se peut qu'elles soient programmées sur des modules météo différents.

Enfin, nous avons installé un bouton « remise à zéro » de la programmation qui supprime tout planning déjà existant. Et un bouton « valider » qui crée la programmation dans la base de données.



Figure 17 Diagramme de séquence de programmation.html

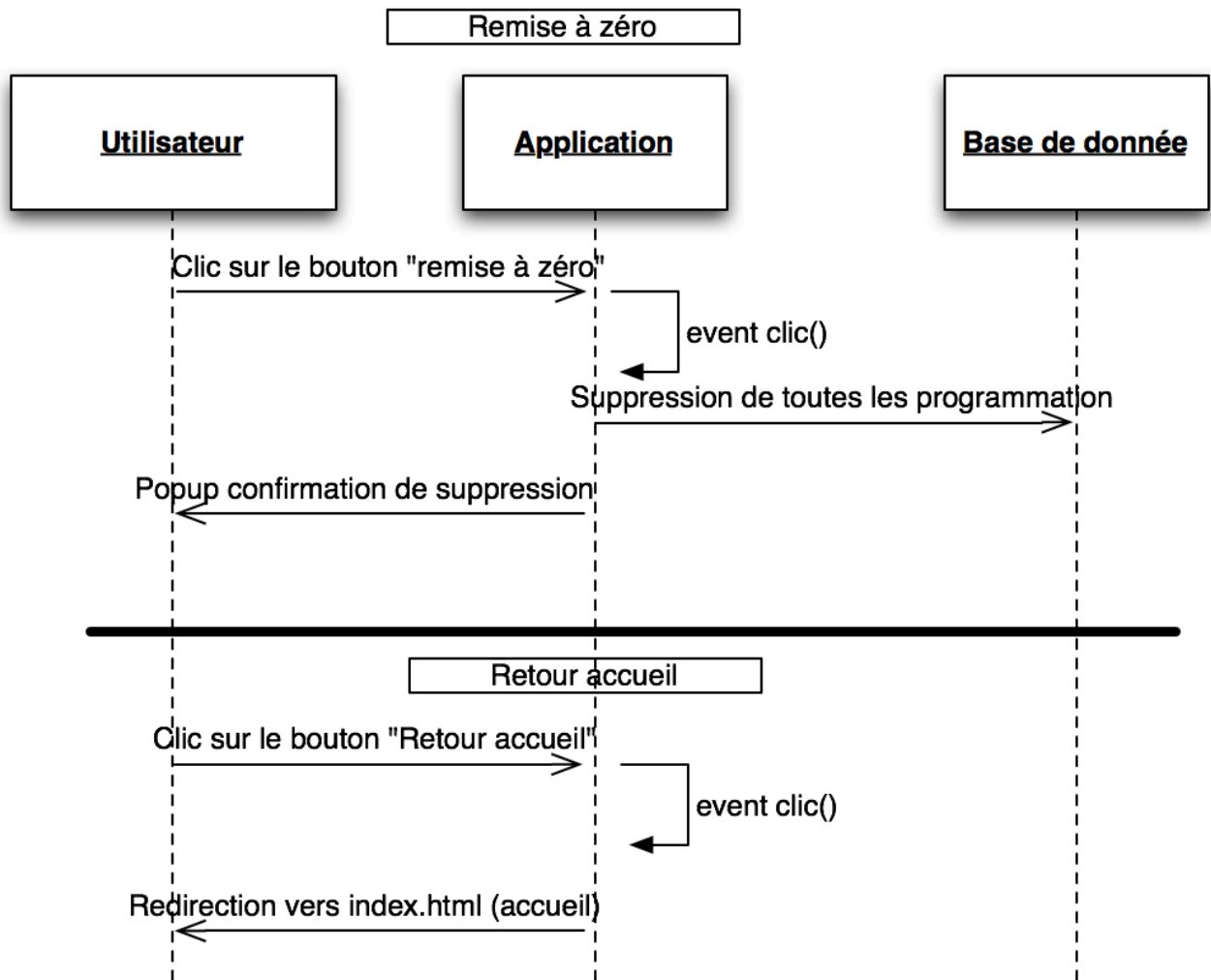
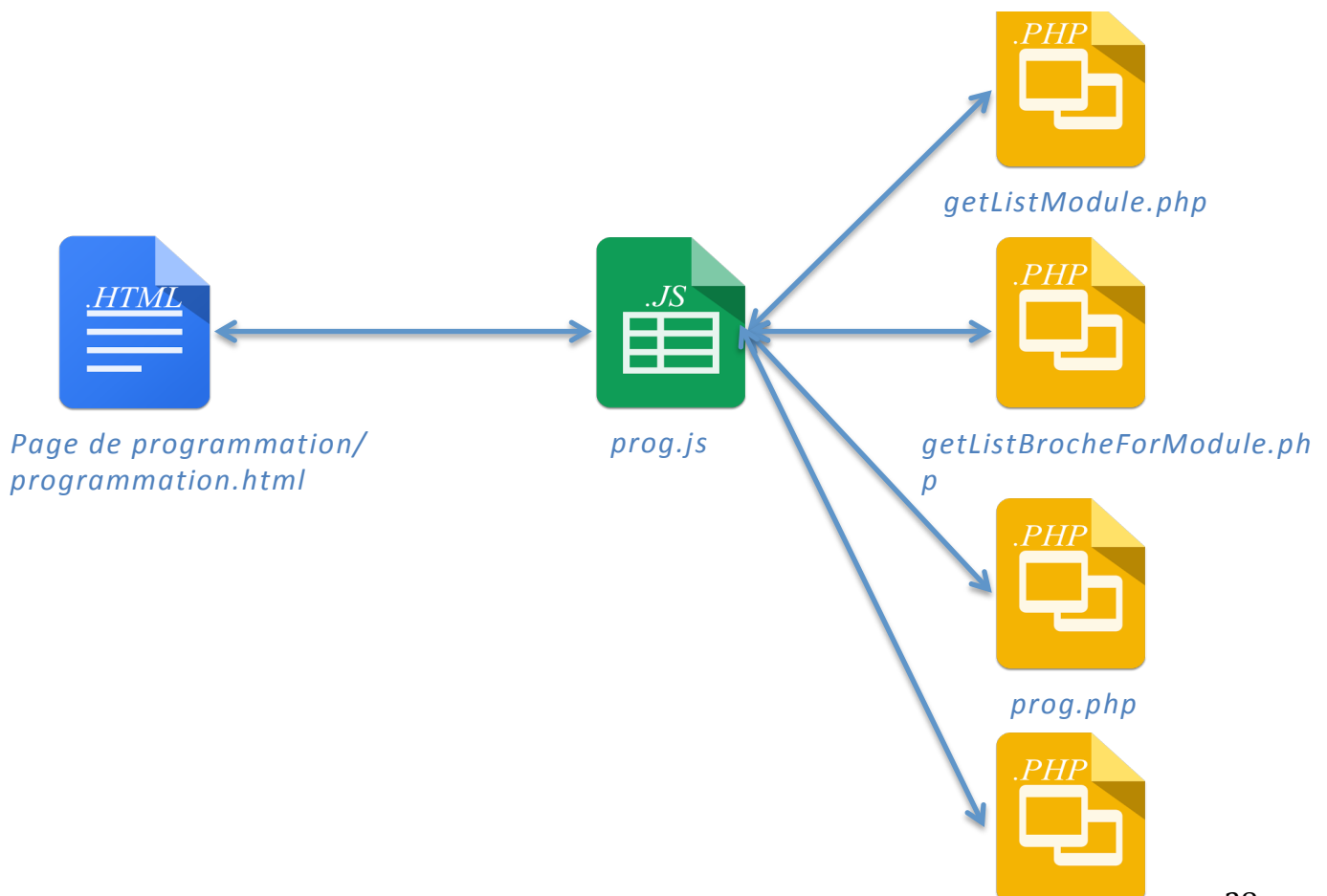


Figure 19 Diagramme de séquence de programmation.html



## Schéma de la base de données

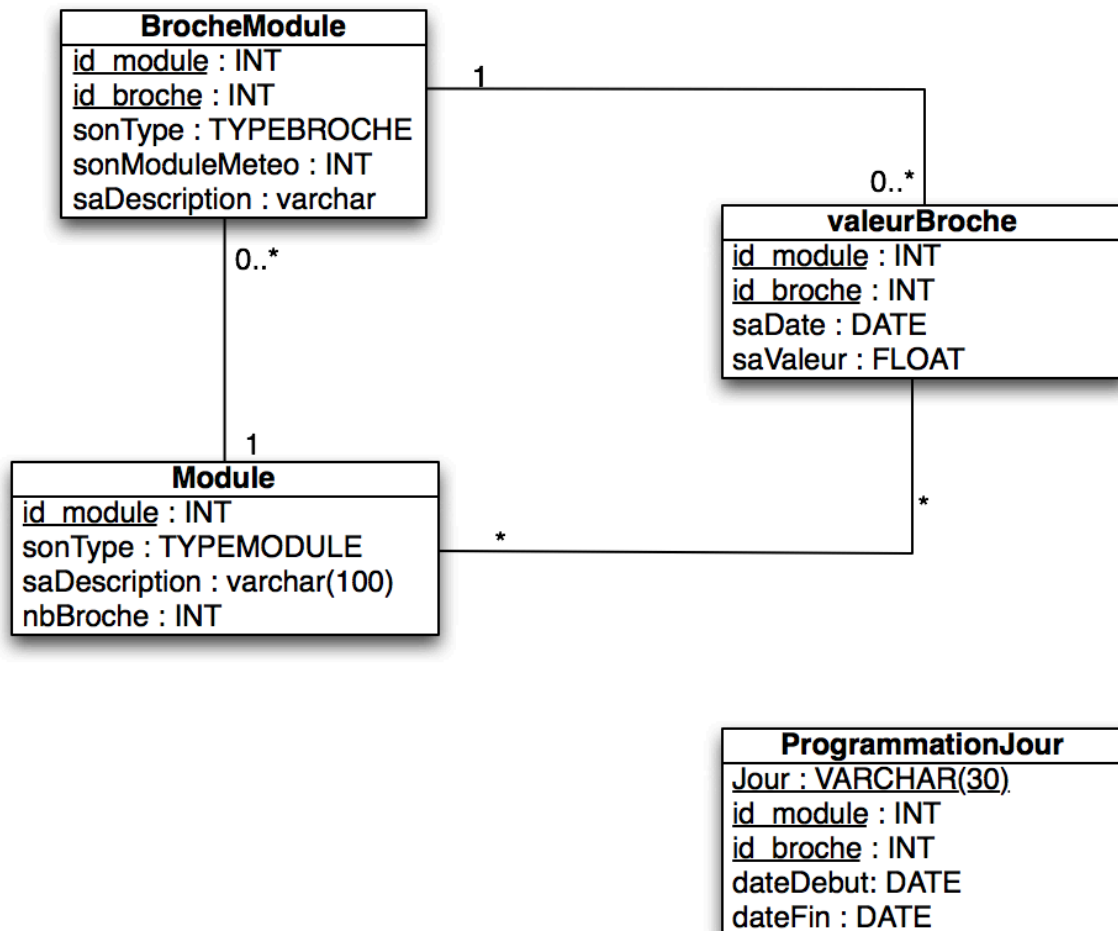


Figure 20 Schéma de la base de données

Voici le schéma de notre base de données, elle est organisée de façon à facilement retrouver les informations voulues.

- **Module** : C'est ici que sont créés tous les modules.
  - TYPEBROCHE : cette énumération peut prendre 3 valeurs différentes → CAPTEUR, INTERRUPTEUR ou AUTRE.
- **BrocheModule** : Création des broches pour chaque module.
  - TPEMODULE : énumération pouvant prendre la valeur METEO ou COMMANDE.
- **Valeur broche** : Table contenant toutes les valeurs de chacune des broches.
- **Programmation** : Table contenant la programmation de chacune des broches.

## Les axes d'améliorations

### La gestion des commandes manuelles :

Pour le moment, l'application fonctionne grâce au scheduler qui fait appel à la page 'gestionOuvertureVanne.php'. Cette page est prioritaire sur la page Commandes Manuelles. Lorsque l'utilisateur sera amené à utiliser la page de commandes manuelles, il sera nécessaire d'interrompre le scheduler.

### La visualisation de la programmation :

Nous souhaiterions mettre en place un système permettant à l'utilisateur de visualiser les informations de programmation se trouvant dans la base de donnée.

### Le nombre d'arrosage :

Actuellement, il n'est pas possible de programmer plusieurs arrosages le même jour, cela est dû à l'architecture de notre base de données. Pour pallier à ce problème, nous avons pensé à quelques axes d'améliorations mais rien de viable pour le moment.

### Gestion du changement d'heure :

Lors de la mise en place du serveur, nous avons dû programmer l'heure. Nous ne savons pas si, lors du changement d'heure, le serveur se mettra à jour automatiquement. Si cela n'est pas le cas nous devons donner à l'utilisateur la possibilité de le faire.

## Annexes

Pour la base de données, le fichier de création ainsi qu'un jeu de test sera fourni. Pour une explication plus précise de chacun des scripts php, un en-tête est placé au début. Quelques commentaires sont présents sur certaines fonctions pour plus de détail.