Projet CIR2

GESTION DE DONNÉES D'OBJETS CONNECTES

Retours sur la présentation client

michael.aron@isen-ouest.yncrea.fr
yann.le-ru@isen-ouest.yncrea.fr
didier.le-foll@isen-ouest.yncrea.fr
fabienne.provost@isen-ouest.yncrea.fr

Mis à jour le 15/05/2017

Sommaire

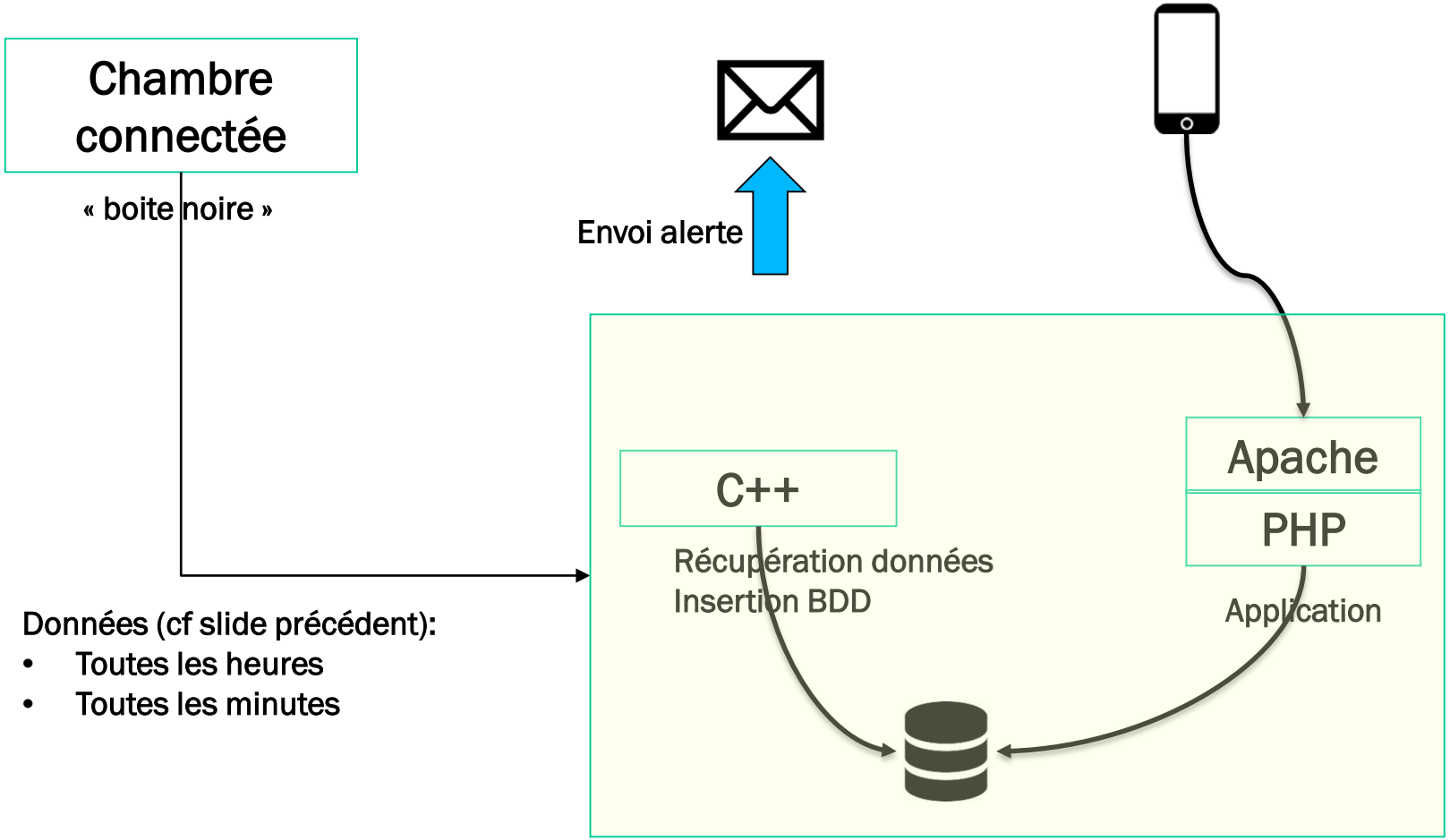
- Général
- Organisation
- Architecture
- Base de données : MCD
- Web
- C++

- Se concentrer sur les fonctionnalités exprimées dans le cahier des charges
- Prévoir une gestion par lots de façon à avoir un système opérationnel :
  - Version 1 : minimum n'incluant que les fonctionnalités de base (design minimal)
  - Version 2 : ajout de fonctionnalités présentes dans le cahier des charges + design amélioré
  - Version 3 : ajout des fonctionnalités d'administration, optionnelles et « cosmétiques »

Organisation

- Environnement de développement :
  - Attention aux environnement hétérogènes (cohabitation windows – linux), et aux versions différentes des librairies
  - BDD sur le serveur de pré-production ou en local
- Serveur de pré-production :
  - méthode utilisée pour la mise à jour des codes sur le serveur
    - Soit utiliser un client SFTP (attention à ne pas écraser le code de l'autre !) et en ligne de commande Linux un SCP
    - Soit utiliser un dépôt GIT (sauvegarde des versions postées mais possibilité d'avoir des conflits)

Architecture



Données (cf slide précédent):

- Toutes les heures
- Toutes les minutes

- Alertes
  - Côté serveur pour l'envoi de mails, de SMS…
  - Côté client pour affichage sur l'application web

- Alerte côté serveur (recommandé)
  - Gestion à partir du programme C++ qui récupère les données : il peut ainsi également prendre en charge le traitement et l'envoi des alertes (directement en C++ ou utilisation d'une commande system pour lancer un script ou un programme PHP)

- Alerte côté client (recommandé)
  - Utilisation d'ajax + un timer pour l'affichage (solution la plus simple, de type « POOL »)
  - Utilisation de websocket (Qt, nodeJS…) (solution de type « PUSH »)

Base de données

- Mettre les données (informations de connexion sur les chambres connectées, valeur des seuils…) dans la base de donnée (et non en « dur » dans le code C++ ou PHP…)
  - IP + port de la pièce où se connecter
  - Seuils d'alertes. Ces seuils peuvent être différents suivant la pièce
  - Log : garder une trace des alertes émises dans la BDD

- Une maquette web doit présenter l'IHM au client
- Responsive design (affichage adapté pour tablettes/smartphones)
- Indiquer sur l'IHM l'heure de la dernière mise à jour

- Connexion à la BDD pour récupérer les infos de connexions des pièces et utilisateurs (IP + ports)
- Récupération des données des pièces et des utilisateurs avec la classe QWebSockets : implémentation d'un client
- Parsing du flux avec la classe QJSon
- Traitement des données (conversion des données du capteur MTH02 par exemple) + gestion des alertes
- Envoi des informations à la BDD à l'aide de la bibliothèque MySQL Connector C++

FIN

Des questions?