# ATOS IT Challenge 2015, with the partnership of EEBUS

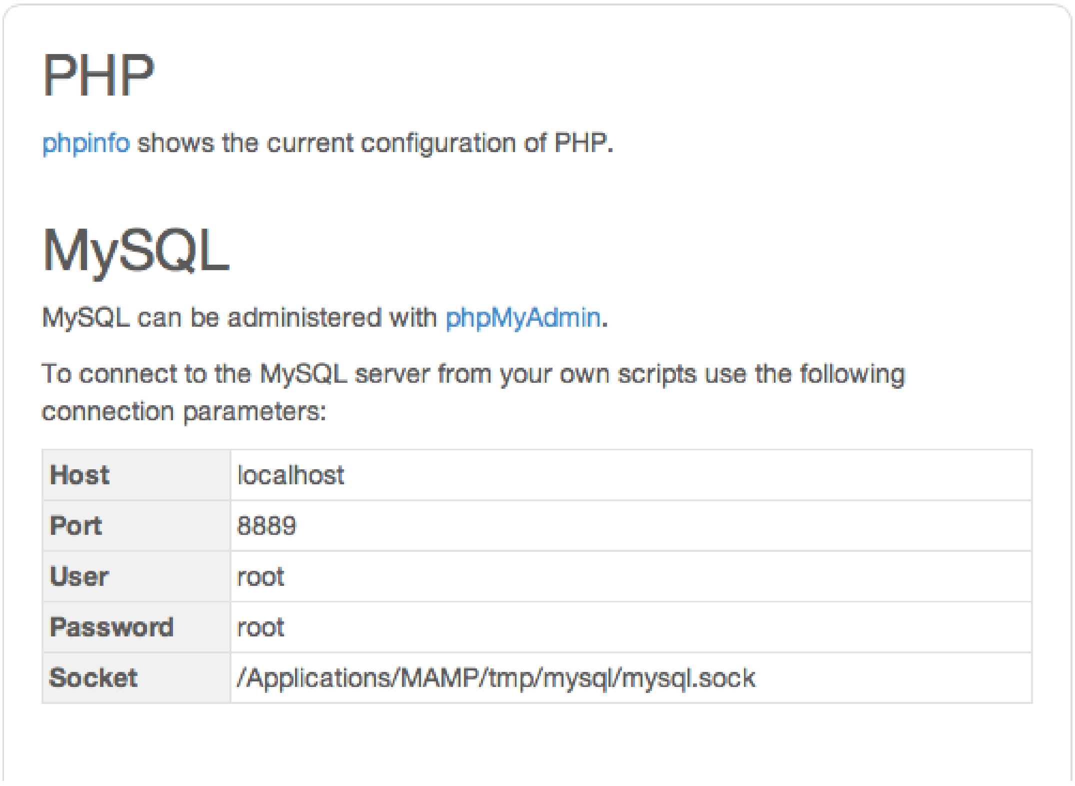
**Idea Name:** EASE - Your Home in Your Hands
**Idea Description:** EASE, as it sounds, aims to facilitate people's interactions with their homes.It provides the user with the most fitted workflow to perform a task by taking account of the abilities of home devices and their ecological impact. By using habit learning, EASE also filters and forecasts tasks and their solutions. As the solution been proposed, EASE can suggest the user to check for the best smart devices sold in the market in terms of efficiency and connectivity; and business services such as food delivery and online purchasing to give more options and flexibility.

This prototype only focuses on the generation of proper workflows, and the launch of tasks.

**Team Name: EASE**
**Team Members:** RONG Han, TALEB Aladin, HE Lei

**How to build app:**
The server is not available online, you have to launch it on your own computers. Here is a short tutorial.

1. You need an app-server-database platform like MAMP(for Mac), LAMP(for Linux) or WAMP(for Windows).
2. Create a database with xAMP (it can be called EASE), then get the login informations on the webstart page



## PHP

phpinfo shows the current configuration of PHP.

## MySQL

MySQL can be administered with phpMyAdmin.

To connect to the MySQL server from your own scripts use the following connection parameters:

| | |
|---|---|
| **Host** | localhost |
| **Port** | 8889 |
| **User** | root |
| **Password** | root |
| **Socket** | /Applications/MAMP/tmp/mysql/mysql.sock |

you have to configure your database connection in our source code:
atos-EASE/EASE-server/config/connections.js
find the code line :

```
 localMysql: {
    adapter: 'sails-mysql',
```

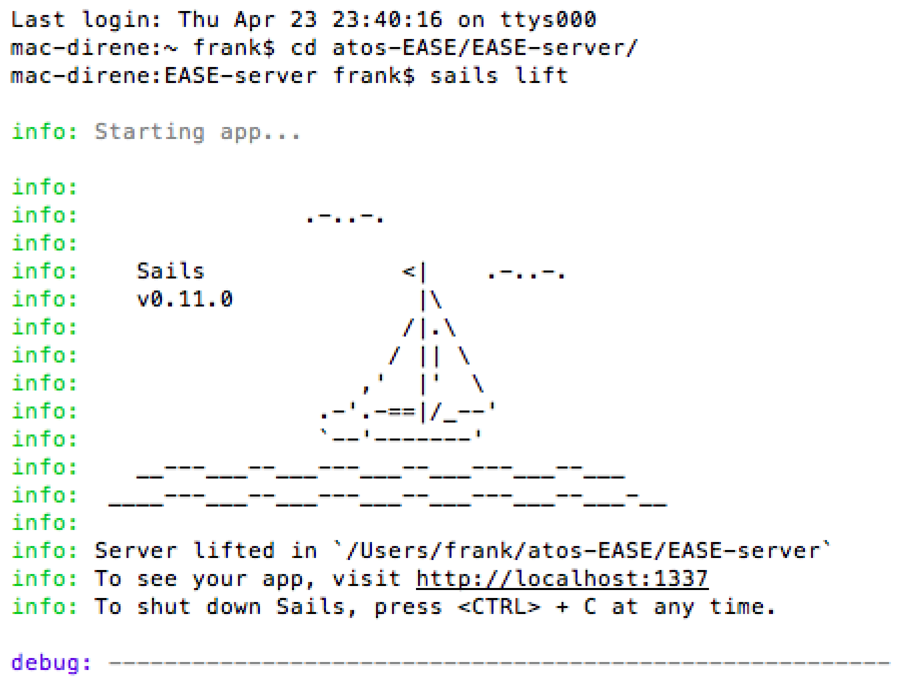
*host: 'localhost',*
*port: 8889,*
*user: 'root',*
*password: 'root',*
*database: 'EASE'*
*},*
and replace it with your information of xAMP, just be sure the database is MySQL.

3. You need the Sails framework( Available on the official website of Sails **sailsjs.org)** on your computer who serves as Local Server for the application.

4. Once the sails framework is installed on your computer, use the terminal to get into the root of our source code, and then change to the directory : EASE-server.

The whole path looks like this: *whereYouPutOurProject/atos-EASE/EASE-server*

5. Before starting the server, be sure that MAMP is on. Start the server by typing sails lift in the command line

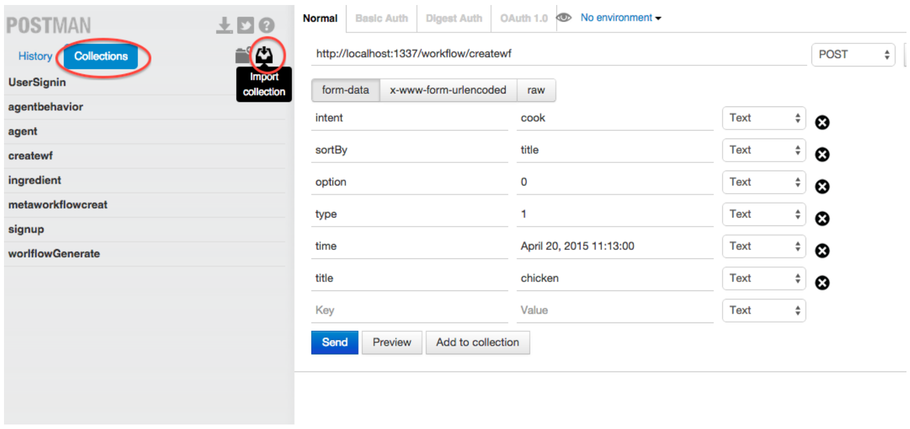


```
Last login: Thu Apr 23 23:40:16 on ttys000
mac-direne:~ frank$ cd atos-EASE/EASE-server/
mac-direne:EASE-server frank$ sails lift

info: Starting app...

info:
info:                .-..-.
info:
info:    Sails              <|    .-..-.
info:    v0.11.0             |\
info:                       /|.\
info:                      / || \
info:                    ,'  |'  \
info:                 .-'.-==|/_--'
info:                 `--'-------'
info:    __---___--___---___--___---___--___
info:  ____---___--___---___--___---___--___-__
info:
info: Server lifted in `/Users/frank/atos-EASE/EASE-server`
info: To see your app, visit http://localhost:1337
info: To shut down Sails, press <CTRL> + C at any time.

debug: --------------------------------------------------------
```

6.Insert some sample data into the database to have fun with EASE: to do this part, please refer to the file : data_ import_postman.json in atos_EASE directory.

Example for inserting collection of request on postman:

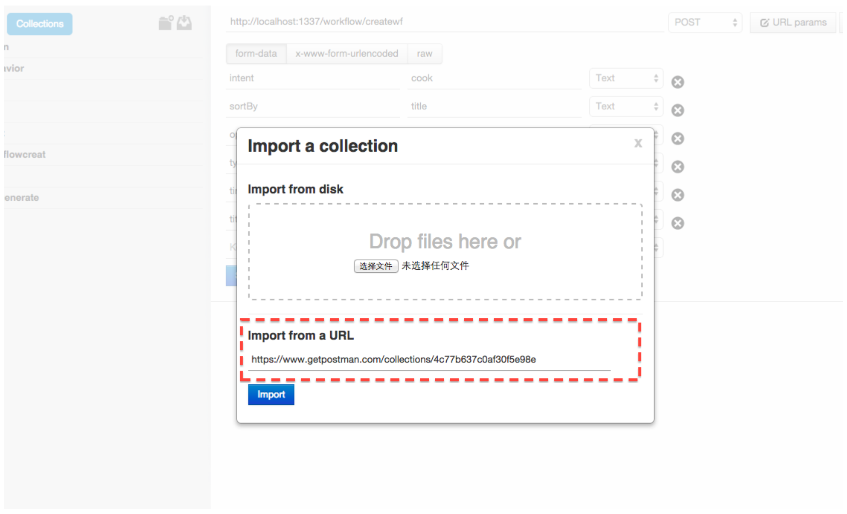
a. Get into the collection tab and click on the icon to import collection
b. A window will pop up demanding a file or url . In our case you should put each url in data_import_postman in the field for url.
c. The importation works if you got the same thing as shown in the window
d. You will have a new signup collection tab



UserSignin
agentbehavior
agent
createwf
ingredient
metaworkflowcreat
signup
worlflowGenerate

http://localhost:1337/workflow/createwf — POST

| | | |
|---|---|---|
| intent | cook | Text |
| sortBy | title | Text |
| option | 0 | Text |
| type | 1 | Text |
| time | April 20, 2015 11:13:00 | Text |
| title | chicken | Text |
| Key | Value | Text |

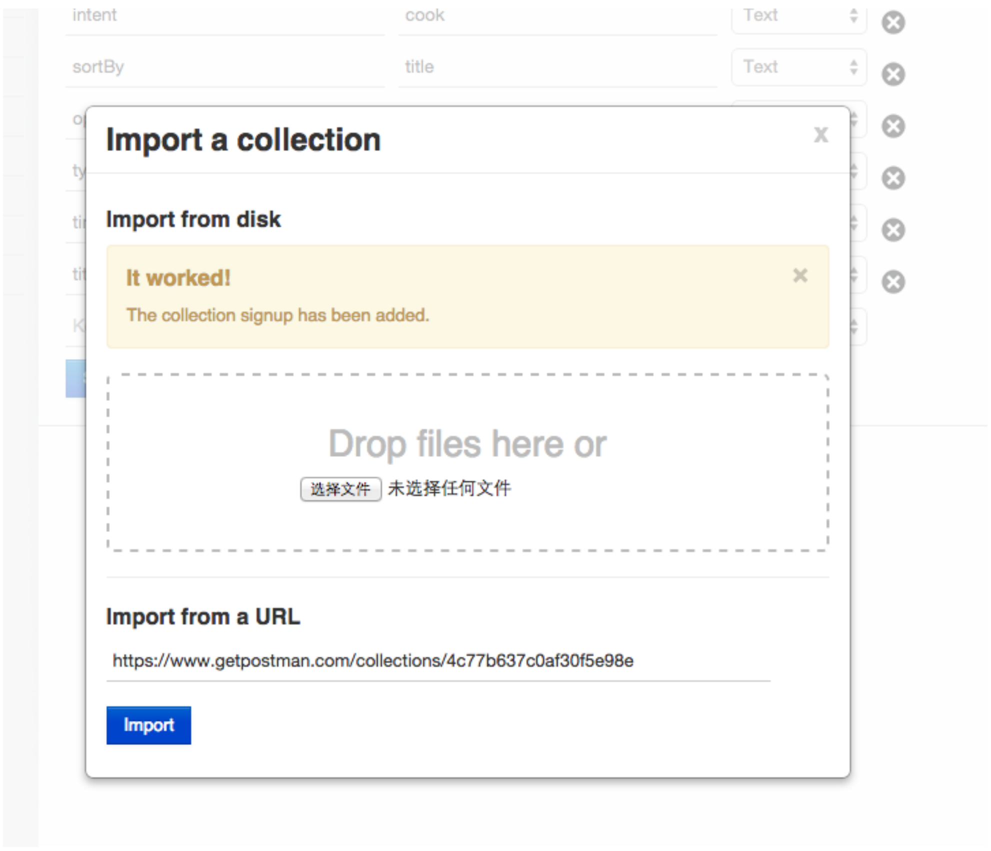


**Import a collection**

Import from disk

Import from a URL

https://www.getpostman.com/collections/4c77b637c0af30f5e98e


Import a collection

Import from disk

It worked!

The collection signup has been added.

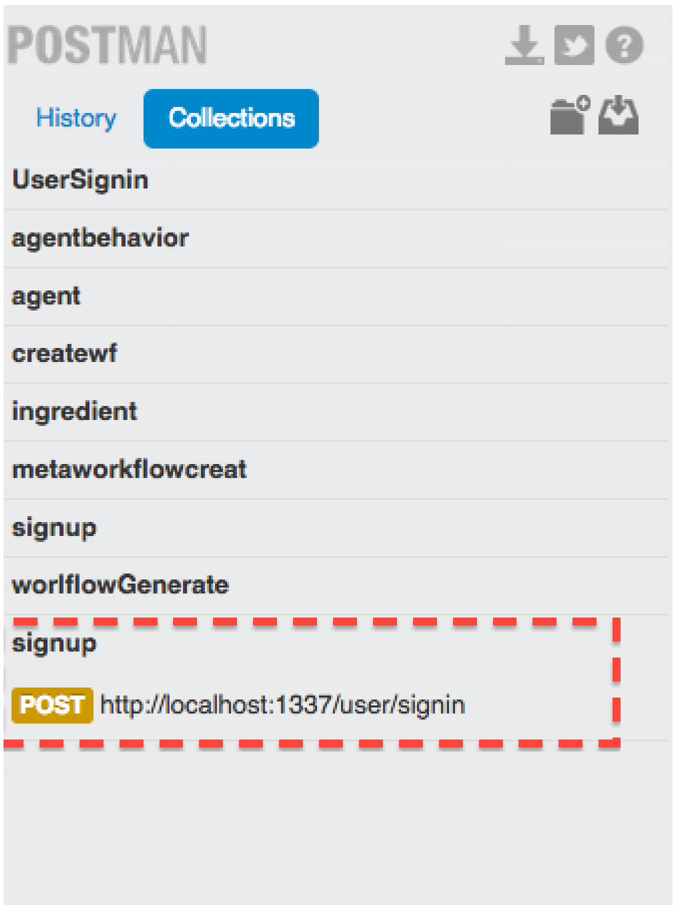
Drop files here or

选择文件 未选择任何文件

Import from a URL

https://www.getpostman.com/collections/4c77b637c0af30f5e98e

Import



UserSignin

agentbehavior

agent

createwf

ingredient

metaworkflowcreat

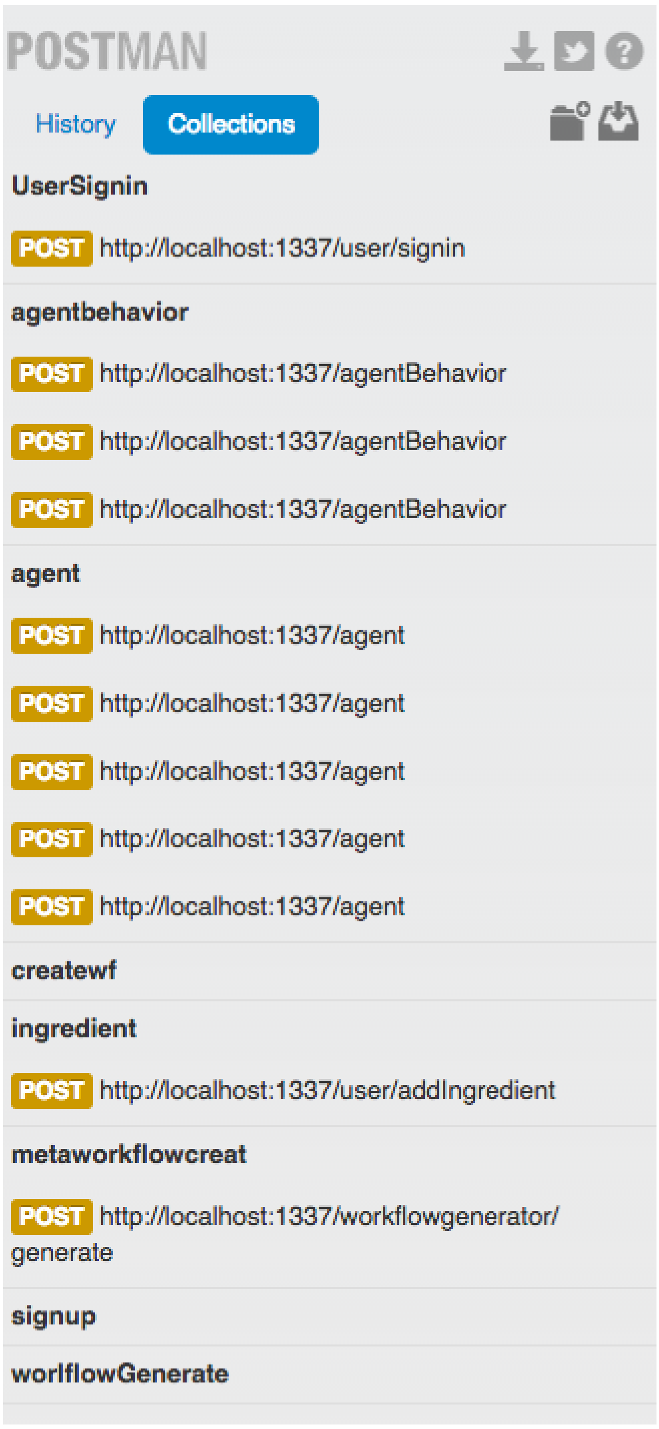
signup

worlflowGenerate

signup

POST http://localhost:1337/user/signin

7. At the end of importing collections, you will have 8 collections shown like in the picture below, send some of the requests following the guide (there is an order to follow) in *data_import_postman.json*



UserSignin
POST http://localhost:1337/user/signin
agentbehavior
POST http://localhost:1337/agentBehavior
POST http://localhost:1337/agentBehavior
POST http://localhost:1337/agentBehavior
agent
POST http://localhost:1337/agent
POST http://localhost:1337/agent
POST http://localhost:1337/agent
POST http://localhost:1337/agent
POST http://localhost:1337/agent
createwf
ingredient
POST http://localhost:1337/user/addIngredient
metaworkflowcreat
POST http://localhost:1337/workflowgenerator/generate
signup
worlflowGenerate

8. Then it’s time to simulate EASE and cooking devices. With the recipe we offer, you need 4 different devices: oven, ricecooker,ricecooker2,microwave. We simulate them with 4 different HTML page. You can get access to them by typing the proper URL in your browser:
http://localhost:1337/ricecooker.html
http://localhost:1337/oven.html

These pages are connected with the server by socket, **open them all** before you start a task in an application, especially if you don’t know which task is done by which agent. Once a task begins, the corresponding device in the HTML page will start and show the time left before the end of this task.

**Platforms:** iOS
**Devices supported:** iPhone with iOS 8.2 or superior. Tested mainly on iPhone 6 with OS 8.2

**Architecture vision:** There are four main roles existing in our architecture: app, server, smart devices and of course users. App is an interface between users, server and the smart devices. It not only affords propositions to users when they want to do something but also controls the connected smart devices to carry out some workflows. Server will do the filter job or problably some calculations and communicate with users and devices. Devices normally execute the tasks and send feedbacks to users regularly.

**Uses cases:** Generally speaking, the most complex procedure we meet often at home in our life is to cook something. Compared to other workflows, a kitchen workflow has more steps, including the preparations and different phases of cooking. Otherwise a workflow like “Make a coffee”, “Turn on the air-conditioner” or “start the laundry” seems to be rather simple to carry out. Hence, we often cite kitchen use cases in our application to present the main functionnalites.
Here is an example: A user wants to cook a chicken at 18h00. To start with, he types “cook a chicken at 18h00”. The application will propose several different ideas to cook a chicken, which can be sorted according to some parameters. App will check of course whether the workflow could be done with using the devices the user possesses. After the user chooses one, app will see if this workflow is possible to be done around the wanted time according to the agents’ timetables. Once the workflow is validated, the user could follow the steps shown in the app to achieve the goal. If all the steps could be done by smart devices, users could even launch them remotely and the workflow will have be done by the time you wanted.
**Does the app use any external API?. If yes, which one?**
The app uses the Wit.ai API (https://wit.ai) to do the natural language processing. The algorithm is run on their own server.

**Does the app use any open source library? If yes, which one?**
Server-side : We used Sails.js, an open source library which allowed us to use a MVC design pattern.
It’s based upon other open source libraries such as

- Node.js : the main server
- Socket.IO : to communicate via socket
- Waterline : a middleware for database
- Blueprint : to easily make REST API
- Grunt
- etc …

Client-side : The iOS App contains many open source controls, most of which are listed in the cocoapod file.

- AFNetworking
- Socket.IO-Client - This library allows the App to communicate with the server via Websocket. Unfortunately, it is still buggy.
- MZFormSheetController
- RESideMenu
- XLRemoteImageView
- JTCalendar

- ...

**Media material**
Here is a video to setup the webserver, with a demo of the app
https://www.youtube.com/watch?v=zrJdpKEZLUo&feature=youtu.be