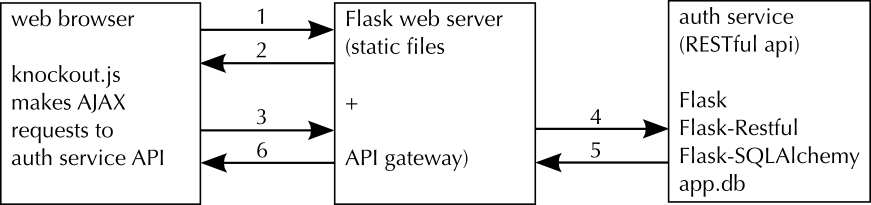
**Main components**

**Main components and data flow**

[](https://gitlab.com/financial/financial/uploads/ea3d950f4b23c0077e5882c8f2022ace/concepts1.png)

1. Web browser opens static web page at <http://localhost:5000/>
2. Flask response with web page (base.html), and all static files (JavaScript, CSS, etc...)
3. JavaSript code in browser makes AJAX request to API gateway at address <http://localhost:5000/api/v1/>
4. Flask proxies call to auth service <http://localhost:5001/auth/>
5. Auth service reponse returns to proxy.
6. Auth service response returns to js code.

# Install, Run and testing

## General security considerations

For testing and demo usage it is normal to request services and web server via http (unencrypted). To protect from data interception web server and services should be accessible only via https encrypted connections.

Also ADMIN\_PASSWORD and ENCRYPTED\_TYPE\_SECRET\_KEY should not be setup in config.py. Instead these can be read on system startup from terminal, entered by system operator.

## Install

* Get fresh 2.7 Python <https://www.python.org/downloads/windows/>
* Install all components (pip required).
* Run [install.bat](https://gitlab.com/financial/financial/wikis/install_bat)

## Run web server, services and tesing SMTP server

* Run [auth\_service.bat](https://gitlab.com/financial/financial/wikis/auth_service_bat) to start authentication service.
* Run [account\_service.bat](https://gitlab.com/financial/financial/wikis/account_service_bat) to start accounts service.
* Run [testsmtp.bat](https://gitlab.com/financial/financial/wikis/testsmtp_bat) to start testing SMTP server to receive emails from Flask
* Run [web.bat](https://gitlab.com/financial/financial/wikis/web_bat) to start Flask web server.
* Open in a browser page <http://localhost:5000/>

## Run unit tests

to run unit tests Run test\_auth\_service.bat and test\_accounts\_service.bat

**Files**

• .env directory

• account\_service.bat

• auth\_service.bat

• install.bat

• requirements.txt

• test\_accounts\_service.bat

• test\_auth\_service.bat

• testsmtp.bat

• web.bat

**1. Frontend code (Works in browser)**

Components used:

* [Twitter bootstrap](http://getbootstrap.com/) - main page layout, navbar, forms layout, etc...
* [knockout](http://knockoutjs.com/) - dynamic binding of JavaScript data with HTML elements
* [Sammy.js](http://sammyjs.org/) - client side routing (switch of pages on URI change)

**2. Flask web server**

**Files description:**

* **app/templates/base.html** - html page template (shown when <http://localhost:5000/> loaded in browser)
* **app/static/js/knockout-3.4.0.min.js** - knockout library
* **app/static/js/base.js** - main code of frontend
* **app/static/js/bootstrap.min.js** - twitter bootstrap library
* **app/static/js/sammy.min.js** - sammy.js library
* **app/static/js/jquery.min.js** - jqyery library
* **app/static/fonts/glyphicons-halflings-regular.ttf** - font of twitter bootstrap
* **app/static/fonts/glyphicons-halflings-regular.woff2** - font of twitter bootstrap
* **app/static/fonts/glyphicons-halflings-regular.svg** - font of twitter bootstrap
* **app/static/fonts/glyphicons-halflings-regular.woff** - font of twitter bootstrap
* **app/static/fonts/glyphicons-halflings-regular.eot** - font of twitter bootstrap
* **app/static/css/bootstrap-theme.min.css** - twitter bootstrap CSS
* **app/static/css/base.css** - local CSS of base.html
* **app/static/css/bootstrap.min.css** - CSS of twitter bootstrap
* **app/views.py** - main code of Flask web server (route '/' lodads base.html, and route '/api/v1/' proxies API calls to auth api)

**3. Auth service**

**RESTful methods**

| HTTP method | Authentication required | URI | Action | |
| --- | --- | --- | --- | --- |
| GET | Y | <http://localhost:5001/auth/> | | returns user id if username/password correct, [401 Unauthorized](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#4xx_Client_Error) in other case |
| GET | Y | <http://localhost:5001/auth/profile> | returns user profile if authenticatd | |
| POST | N | <http://localhost:5001/auth/register-student> | register student | |
| POST | N | <http://localhost:5001/auth/register-parent> | register parent | |
| POST | N | <http://localhost:5001/auth/forget-password> | send email with token to reset password | |
| POST | N | <http://localhost:5001/auth/reset-password> | reset password if token correct and not expired | |

**Files description:**

* **services/auth/app.db** - sqlite database of auth service. Tables can be seen by [SQLite browser](http://sqlitebrowser.org/)
* **services/auth/app/models.py** - models of SQLAlchemy ORM. In file described models **User**, **Student**, **Parent**, and many-to-many relation **children**
* **services/auth/app/validators.py** - validators of RESTful input data.
* **services/auth/app/views.py** - RESTful API entry points

## 4. Accounts service

## RESTful methods

| HTTP method | Authentication required | URI | Action |
| --- | --- | --- | --- |
| GET | Y | <http://localhost:5002/acc/> | Get account information (daily limit, balance) |
| POST | Y | <http://localhost:5002/acc/> | Create account if it does not exist |
| GET | Y | <http://localhost:5002/acc/daily-limit-for> | Get daily limit for child |
| PUT | Y | <http://localhost:5002/acc/daily-limit-for> | Set daily limit for child |
| POST | Y | <http://localhost:5002/acc/deposit> | Make deposit to own account |
| POST | Y | <http://localhost:5002/acc/send-money-to> | Send money to child |
| POST | Y | <http://localhost:5002/acc/spend> | Spen money (only for child) |
| GET | Y | <http://localhost:5002/acc/transactions> | Get account transactions |
| GET | Y | <http://localhost:5002/acc/transactions-for> | Get child transactions |

## Detailed desription of files

• services/accounts/run.py

• services/accounts/config.py

• services/accounts/test\_app.py

• services/accounts/test\_config.py

• services/accounts/app.db - sqlite database of accounts service. Tables can be seen by SQLite browser

• services/accounts/app/\_\_init\_\_.py

• services/accounts/app/models.py

• services/accounts/app/views.py

• services/accounts/app/validators.py