# Zia

## Projet pour le module C++ - HTTPd

Giron David thor@epitech.net

*Abstract: The goal of the Zia project is to create an HTTP Server. This server will be able to handle classic HTTP documents and pages requests, CGI execution, etc... The server must be written in C++, with interoperable modules support.*

# Table des matières

## I Introduction

The `C++ - HTTPd` Knowledge Unit (KU) is the natural following of the first semester `C++ - Windows` KU. This KU is composed of only one large scale project : The `Zia`, a modular http server written in `C++`.

All the knowledge you acquired from the previous KU will be of great help as will be every abstraction you already wrote. This project will be your last object-oriented conception and implementation in `Epitech` (out of your end of studies project, of course).

It is important to point out that you must treat each of your conception diagrams, as to write clean and professional code - two of the most important informations in this document.

Anyway, conception and code are not the very heart of `Zia`. You will overrun a challenge none of you ever faced in your scholarship : Team up with your whole class! More about this below...

## II Project lifetime

The `Zia` lasts the whole second semester for 3rd year students. It is split in the following 5 main steps :

- The protocol defense
- The API defense
- The API election
- The pre-final-defense follow up
- The final defense

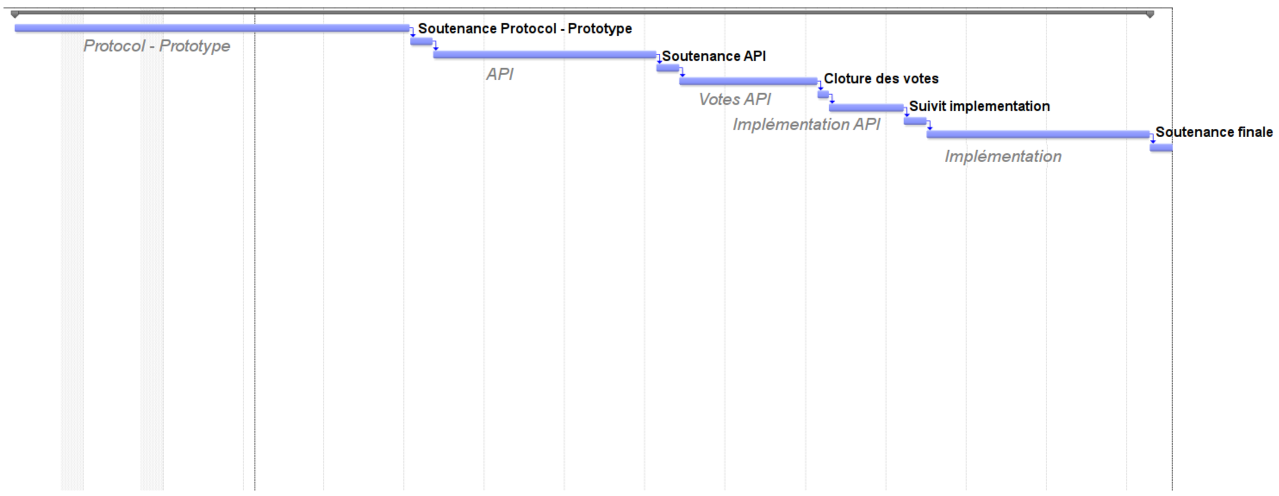

Fig. 1 – `Timeline`

Each one of these items will be discussed in details below.

# III Common Parts

## III.1 Protocol

The `Zia` **MUST** be an exhaustive implementation of the `HTTP/1.1` protocol, strictly following the `RFC 2616`, with the exception of proxies support. The `RFC 2616` can be found anywhere on the internet, for instance, here : http://perso.epitech.eu/~koala/ressources/rfc2616.txt.

The first defense will test your knowledge of the `RFC 2616`, but if you wrote your own correct little RFC during the previous `R-Type` project, reading this big one shouldn't be a problem. Refer to the `R-Type`'s subject for more informations about RFCs.

Here is a strictly **NON** exhaustive list of what we consider mandatory to know about the `RFC 2616` :

- The requests structure
- The answers structure
- The Http methods
- The Http answers codes
- . . .

## III.2 Server Configuration

The server must be fully configurable by means of a configuration file. Any software configuration by compilation will **NOT** be accepted and severely retributed.

You can use any format for your configuration file, XML like, Json, Unix like (sections/keys/values), ...

Several notes about the configuration file :

- You **MAY** use an XML or Json parsing library. Before you ask : `Boost::Spirit` is **NOT** allowed
- You **MUST** implement a regular and coherent parser by hand if you choose not to use any parsing library
- You **MUST NOT** use XML or Json libraries for other purpose than parsing the configuration file. This is considered cheating
- The server **MUST NOT** crash if started with a buggy or missing configuration file : You **MUST** set default values!
- The configuration file **MUST NOT** be found with an absolute path
- It **MUST** be possible to reload the configuration file without restarting nor recompiling the server

# IV Modules

As a modular server, `Zia` **MUST** be able to handle modules. Actually, Zia could be seen as an empty shell that must be filled with modules in order to work.

A module is an atomic processing unit with an input and an output. It `MUST` be possible to use multiple modules together in order to create a processing line to handle an http request and create the appropriate http answer.

You Must design a complete application programing interface to handle your `Zia`'s modules. Your API will be evaluated during the API defense.

Each group **MUST** work on its own API. While interaction and mutual help is encouraged, each API must result of the work of one and only one whole group.

Although submiting your API to the election is optionnal, you **MUST** provide one during the API oral defense.

Whichever API is elected, each group **MUST** also provide two mandatory modules for the final defense : The secure connection module and the PHP CGI module. Once these two modules work perfectly, you **SHOULD** add as many other modules as you like in order to raise your final grade.

## V Interoperability

Modules from different groups **MUST** be interoperable. One module from a group **MUST** run seamlessly in another group's server. Recompilation and server restart are **NOT** allowed when adding or removing a module.

This task is particularly complex and requires some reflection, research and discussions between groups. Note that when the API has been chosen, every group will have to use it, regardless of its quality.

Here are a few rules you **MUST** respect :

- Modules **MUST** be able to hook up to any stage of request processing, from connection establishment to page rendering and everything in between.
- The server **MUST** be able to load/unload modules dynamically, which means these operations **MUST** be performable while the server is running.
- Zia's theorem :
  - Let `Z` and `M`, respectively the set of Zias and the set of modules produced by the entire tech3(s) class.
  - Let `S` the set of the subsets of `M` as $\forall s \in S, s \subset M$.
  - Let `B` the set defined as {`OK`, `CRASH`}.
  - Let $R : (Z \times S) \longrightarrow B$, the binary relation between a couple composed of a an element of `Z` and an element of `S` in `B`.
  - Then we have $\forall (z, s) \in (Z \times S), zRs = OK$

## VI Secure Connections Module

The first of the two mandatory modules. The server **MUST** offer the ability to establish secure connections using `SSL` or `TLS`. This feature **MUST** be a module. You are allowed to use `OpenSSL`. More informations at http://en.wikipedia.org/wiki/Transport_Layer_Security.

## VII PHP Interpreter Module

Second of the two mandatory modules, this module **MUST** allow the execution of `PHP` scripts. The scripts **MUST** be run as CGI.

# VIII Developpement constraints

## VIII.1 General

- Git, Mercurial and SVN repositories will be created several hours after the inscription are closed. You **MUST** use one and only one repository of your choice to host your Zia project
- Your Zia **MUST** compile and run with at least one Microsoft Windows **AND** one UNIX systems
- Your Zia **MUST** be multi-threaded
- You **SHOULD** use your previous abstractions
- You **SHOULD** write Unit tests
- You **MUST** write C++ code, C and C+ wont be tolerated

## VIII.2 Libraries

- STL is allowed in both server and modules
- Boost is forbidden in the server but allowed in request processing modules. In case of a modular core, Boost is of course also forbidden in such modules
- Qt **MAY** be tolerated in the server for GUI purposes ONLY and is forbidden in the modules
- Whatever library not clearly allowed is forbidden. Mail your teacher if you have questions

# IX Testing modules and servers

During your defense your server must support these client softwares :

- Telnet
- Lynx $\geq$ 2.8.*
- Firefox $\geq$ 3
- Opera $\geq$ 7
- IE $\geq$ 6
- Siege (soft)
- Siege (strong as hell)

# X Realization steps

## X.1 Protocol/prototype defense

First of the five steps, the Protocol/prototype defense stands for 20% of your final grade. The main goal of this defense is to evaluate the knowledge of your group of the RFC 2616.

The graduation system is quite simple : Each student owns potential points according to the number of students in his group. For instance, in a group composed of 5 students, each student owns 20/5=4 potential points.

Each member of your group is asked a question about the RFC 2616. If the answer is right, the group wins the points. If the answer is wrong or if the student can't answer, the group **LOSES** the points.

Once the questions are over, you're free to show your diagrams to the `Koalas` and discuss your design within the remaining time.

## X.2 API defense

Second step, this defense also stands for 20% of your final grade. Although submitting **YOUR** API for the election is a choice, defending **YOUR** API during this defense is totally mandatory.

## X.3 API election

3rd step, the election will last from the following day of API oral defense to the votes closure. A few important rules to know :

- Submiting your API is a choice
- No support will be provided from the adm except a room, a video-projector and a web interface to vote. You are alone.
- Teachers and assistants are forbidden to interact with you during this period of time. You are alone.
- Maybe the most important rule : The API election is common to the whole France ! Candidates **MUST** communicate with ALL the other towns ! In case the elected group is convicted of ignoring one or several other towns, the group will face retributions.
- Beeing elected prom's API is rewarded with a **LOT** of points during the final defense

## X.4 Follow up

4th step, this follow up will allow us to control your progress on the project and evaluate its quality. The grade stands for 10% of the final grade. It very important that you be very dynamic during this follow up, it's the last step before the final defense !

## X.5 Final defense

The final defense stands for 50% of your final grade. We will not only evaluate your product, but also your personnal `C++` knowledge. Each one of you will spend a small part of the defense time face to face with a `Koala` who will question you about `C++` and conception basics. Take this evaluation seriously : If you can't answer these questions, even if your group scored 15/20 **you will FAIL the HTTPd module**! You've been warned.

`Read the last paragraph one more time.`

# XI Groups and Realization

- This project will require groups from 3 to 6 students.
- Only one grade will be given to each group, each member receiving the same grade, **BUT** your module validation **WILL** probably differ according to your **personnal skills and implication**.
- If the teacher requests it or if more that 50% of the group requests it, the grade can be individualized, to avoid lazy students hiding in working groups.
- Once groups are accepted they are definitive, no modification will be allowed. For any reclamation, please contact your teacher.

## XII Contacts and Updates

- Send your UML diagrams to Francois Carruba for feedback!!!
- Koalas have set up an irc chan available to the students. The server is 'irc.epitech.net', chan "#koala". You're welcome anytime just to say hello or ask any question.
- As usual, you can find useful informations here http://perso.epitech.eu/~koala/ and here http://koalab.epitech.net/.
- Keep an eye on this subject, it may change during the next days.