System Application

MORCSAC

REALISED BY :

KORCHI Youssef

KARKOUB Elwali

SUPERVISOR :

Jean-Christophe le lann

Option : SPID (SLS)

Academic Year : 2016-2017

ABSTRACT

Many users including developers face issues while installing software on their computers, nevertheless developers make efforts to simplify for users this important phase to use their software, on this latter matter our project can be classified.

This project consists of developing a website in order to make it exploitable as a platform for a compiler (online) without having to install any software in your computer, therefore **Software as a service** or **SaaS** approach is the solution adopted in order to solve the issue.

SaaS which is a model of software deployment where an application is hosted as a service provided to users or customers across the internet. By eliminating the need to install and run the application on the user's computer, and especially alleviating the user's burden of software maintenance and support also ongoing operation.

CONTENTS

1) Introduction : 5

a) Problem Statement : 5

b) project objectives : 5

2) Developing language : 5

c) Ruby on Rails: 5

i) Definition : 5

ii) MVC pattern : 5

d) Philosophy of rails : 7

e) REST architecture : 8

3) Project management : 9

f) Agile method : 9

g) Gantt's Diagram : 10

4) Project plan and execution: 10

h) Prototype : 10

i) The execution and development platform: 11

j) Results : 12

5) Project deployment : 14

k) what to know before using the cloud : 14

i) Definition : 15

ii) Cloud models : 15

iii) Transformation to cloud : 16

iv) Choosing the vendor : 17

6) Conclusion : 19

l) Appendix A: Status report 22

m) Appendix B : Login code 27

n) Appendix C : signup code 27

o) Appendix D: Home code 28

Figure Lists

Figure 1:MVC pattern ____ *6*
Figure 2:Rails Convention ____ *7*
Figure 3:REST ____ *8*
Figure 4:Agile method ____ *9*
Figure 5:Gantt's project ____ *10*
Figure 6:Prototype ____ *11*
Figure 7:RubyMine interface ____ *12*
Figure 8:Home Page ____ *13*
Figure 9:Sign_up page ____ *14*
Figure 10: Cloud computing ____ *15*
Figure 11: Cloud models ____ *16*
Figure 12 : cloud providers ____ *18*
Figure 13:Login Code ____ *27*
Figure 14:sign up code ____ *28*
Figure 15: home code ____ *28*

1) Introduction :

This project is undertaken as a mandatory requirement for the course of UV6.1. The aim of this course is to realize a technique /scientific study or a project, also to master and use English in written and oral. In this report there is an overall description of the phases that we have passed through from scratch, also it includes what we have achieved until the moment, knowing that we can always add more features and evolve our project.

a) Problem Statement :

Develop a website to play the role of a platform in order to facilitate hosting software on it from the developer's point of view, and on the other hand using software without having to install them on your local computer. That is developing and using software on the fly (online).

b) project objectives :

The main objective of our project is to develop and build a website, that can provides the possibility to use software online, basing on searches that should be done before leading off to the designing and the developing parts.

The searching and designing part is the sinews of war to succeed in our project, therefore as you will mention while reading our report, we gave it more time and care that we could, than we did for the last phase (developing phase).

2) Developing language :

c) Ruby on Rails:

i) Definition :

Rails is a server-side web application framework written in ruby, and it is a model-view-controller (MVC) framework, it provides default structures for database, a web service and web pages. And before using it you have always to take into consideration the conventions.

ii) MVC pattern :

The following image explains the interactions between the different interveners in this pattern :

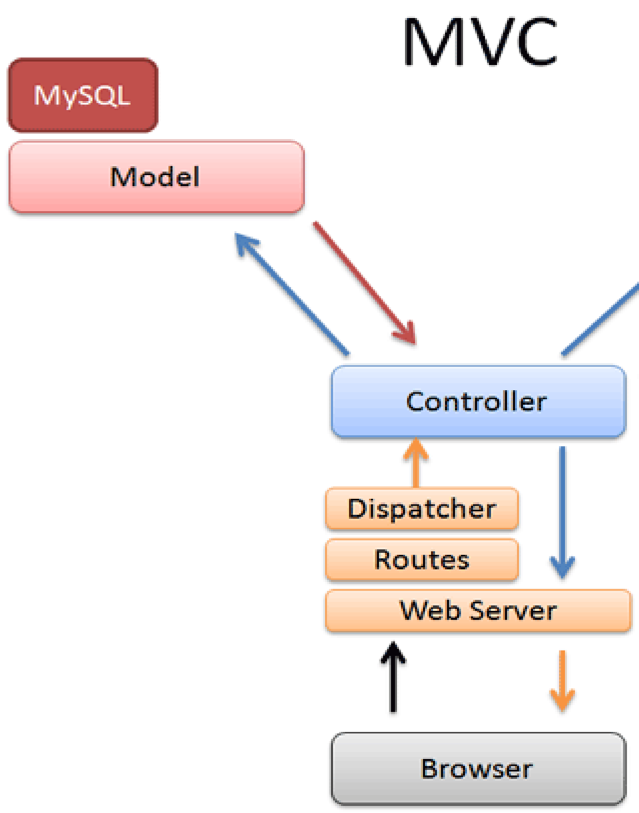


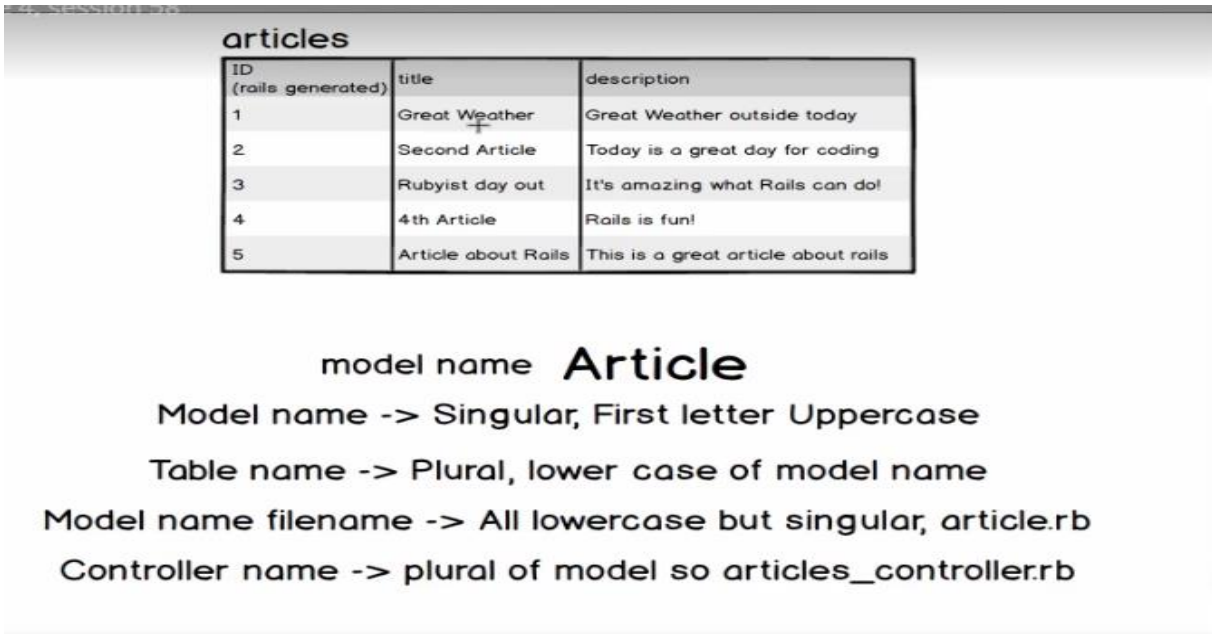
Figure 1:MVC pattern

- **The browser** makes an http request, using a certain URL
- **The Web Server** catches this request, and basing on it, it uses **the routes** to find out which controller to use.
- **Controllers** do the work of parsing user requests, they either render the view requested and redirects it to an appropriate page or the controller gets or add some information to your database passing by the model and then renders the view.
- **Models** are Ruby classes, and they are responsible for communication with the database ,they store and add data basing on the users requests.
- **Views** are what the users see, HTML pages, CSS, and they are reading what the controllers give them.

d) Philosophy of rails :

While coding on rails, you have to take into account the following conventions and principles:

- **Convention Over Configuration (COC) :** also known as coding by convention, is a software design paradigm which seeks to decrease the number of decisions that developers should make, therefore you gain simplicity without losing flexibility, as it is shown below in the image:



articles

ID (rails generated)	title	description
1	Great Weather	Great Weather outside today
2	Second Article	Today is a great day for coding
3	Rubyist day out	It's amazing what Rails can do!
4	4th Article	Rails is fun!
5	Article about Rails	This is a great article about rails

model name **Article**

Model name -> Singular, First letter Uppercase

Table name -> Plural, lower case of model name

Model name filename -> All lowercase but singular, article.rb

Controller name -> plural of model so articles_controller.rb

Figure 2:Rails Convention

- **Don't Repeat Yourself (DRY) :** is a principle of software development to reduce repetition of codes or information, it is applied directly on database schema, test plan and documentation.

- **You aren't gonna need it(YAGNI) :** it is also a principle used in ROR that states that a programmer should not add functionality until deemed necessary, the YAGNI is based on "do the simplest thing that could possibly work" (DTSTTCPW) principle.

e) REST architecture :

THE REST (Representational State Transfer) architecture used by Rails, which is an architectural style for designing distributed systems. it is not a standard but a set of constraints, such having a client/server relationship, and uniforms interface which is the case in Rails, and it is most of time associated with HTTP, also it is a friendly because it can be grasped by users only by reading its syntaxes.

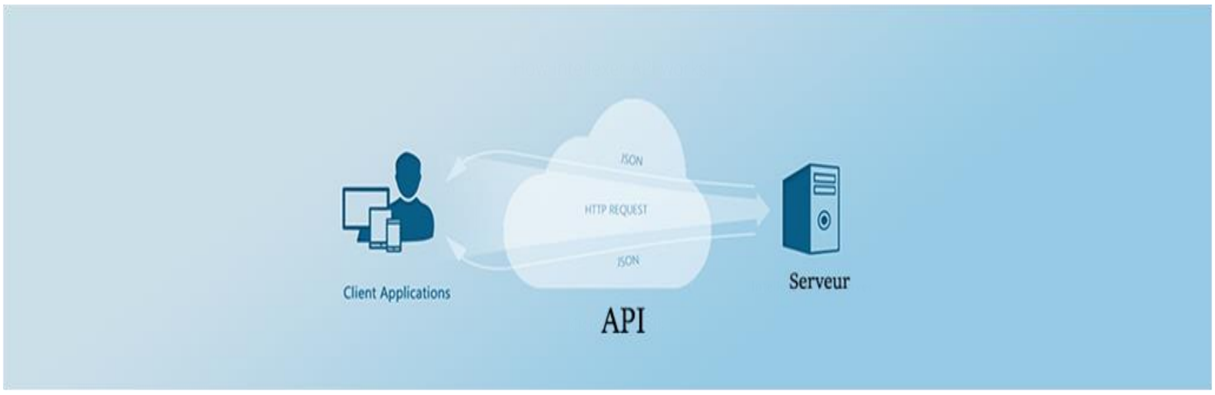


Figure 3:REST

for example when you use the REST:

- **REST API** : GET /account/id/1234
- **Non REST API** : GET /account?id=1234

REST has some principles:

- **Resources** it gives a easily understood directory structure URIs.
- **Representations** transfer JSON or XML to represent data objects and attributes.
- **Messages** use HTTP methods (GET, POST, DELETE, PUT).

- **Stateless** interactions store no client context on the server between requests.

3) Project management :

f) Agile method :

In order to facilitate the dispatch of our tasks, we have used the agile approach which is a methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism since that you have to be realistic while dealing with your tasks also with the time to deliver the finished product. This method focus on keeping code simple, and testing before delivering the tasks tested.

This image represents the principle of the agile method:

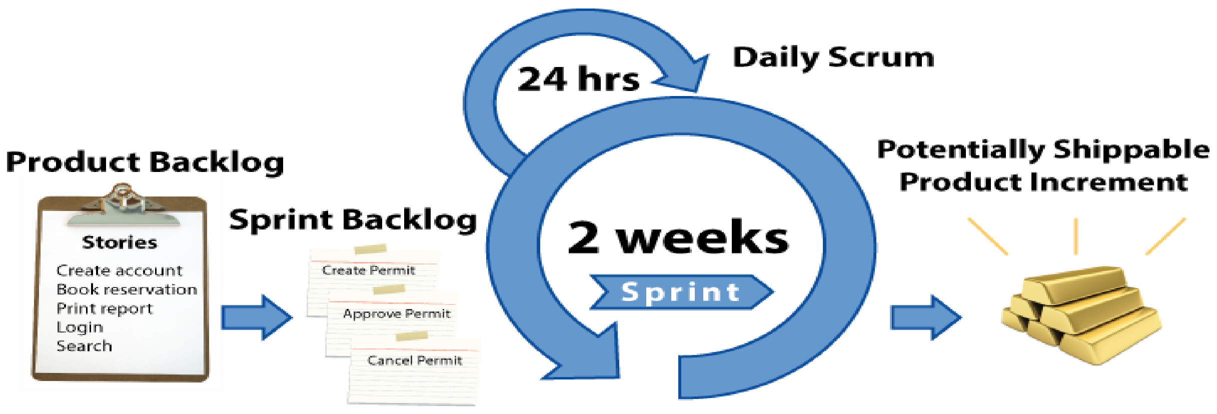
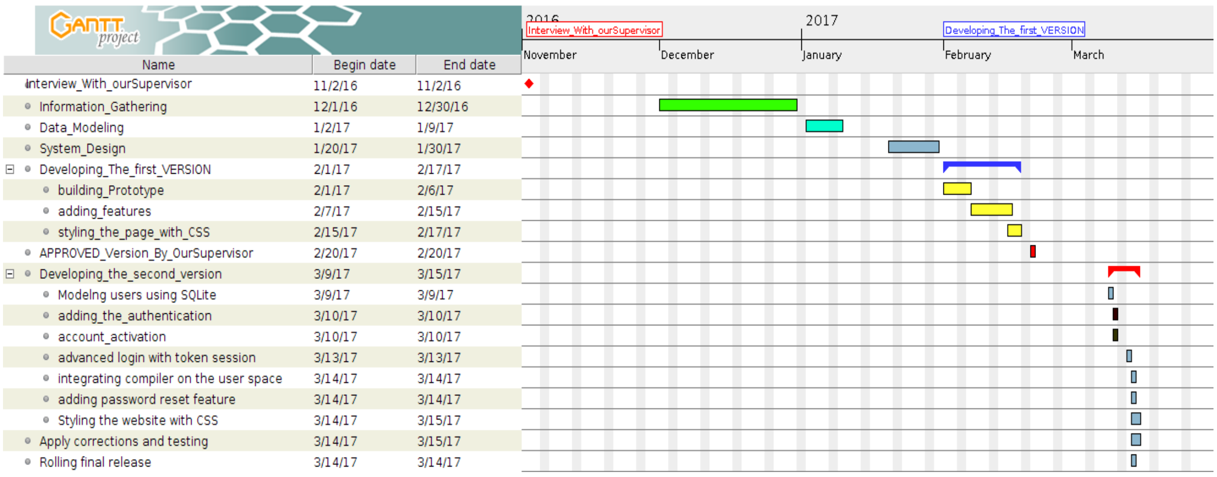


Figure 4:Agile method

dividing tasks into small ones or into sprints as it is named in this approach, helps moving forward easily and surely.

g) Gantt's Diagram :



Name	Begin date	End date
Interview_With_ourSupervisor	11/2/16	11/2/16
Information_Gathering	12/1/16	12/30/16
Data_Modeling	1/2/17	1/9/17
System_Design	1/20/17	1/30/17
Developing_The_first_VERSION	2/1/17	2/17/17
building_Prototype	2/1/17	2/6/17
adding_features	2/7/17	2/15/17
styling_the_page_with_CSS	2/15/17	2/17/17
APPROVED_Version_By_OurSupervisor	2/20/17	2/20/17
Developing_the_second_version	3/9/17	3/15/17
Modelng users using SQLite	3/9/17	3/9/17
adding_the_authentication	3/10/17	3/10/17
account_activation	3/10/17	3/10/17
advanced login with token session	3/13/17	3/13/17
integrating compiler on the user space	3/14/17	3/14/17
adding password reset feature	3/14/17	3/14/17
Styling the website with CSS	3/14/17	3/15/17
Apply corrections and testing	3/14/17	3/15/17
Rolling final release	3/14/17	3/14/17

Figure 5:Gantt's project

4) Project plan and execution:

h) Prototype :

To start the work, our supervisor provided us the following prototype :

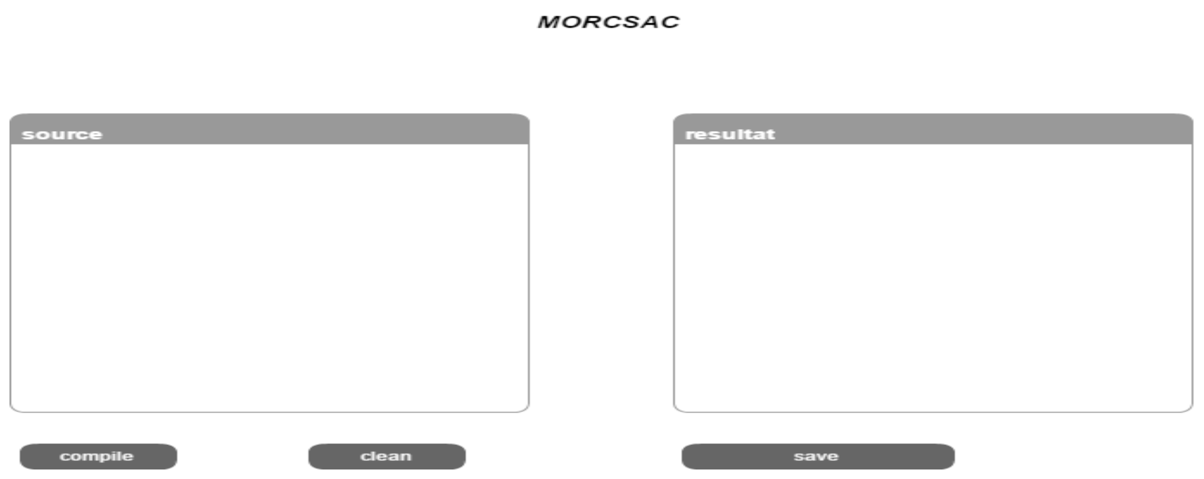


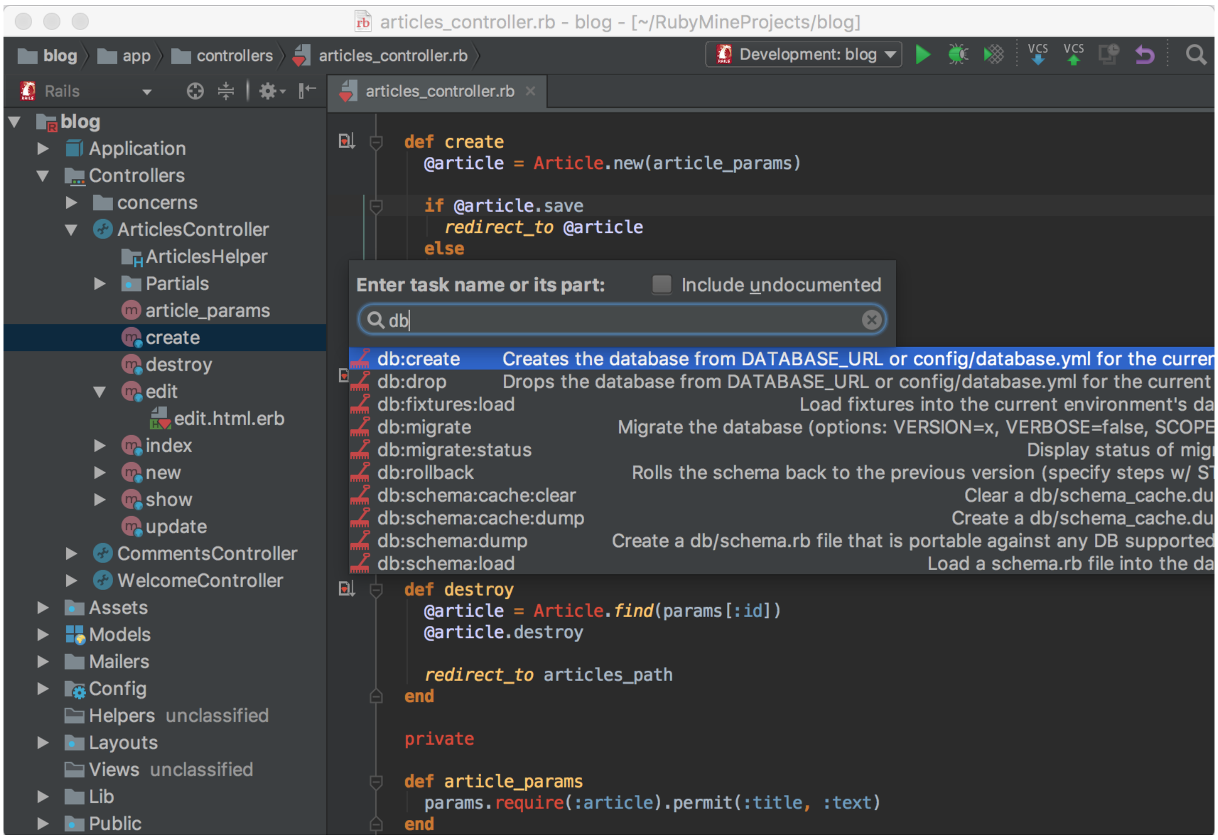
Figure 6:Prototype

This prototype represents the home page of the website

i) The execution and development platform:

For developing and implementation, we had plenty of choices in terms of software to use; after searching in forums for the best one to start with, and based on members opinions we chose to develop with RubyMine.

RubyMine is a cross-platform IDE that supports RUBY, and Ruby on Rails and it is dedicated for web development, moreover it is known as the most intelligent IDE for Ruby on Rails, also it gives you the possibility to use a terminal and you may connect your projects with Github.



```
def create
  @article = Article.new(article_params)

  if @article.save
    redirect_to @article
  else
```

Enter task name or its part: Include undocumented

db

```
db:create             Creates the database from DATABASE_URL or config/database.yml for the curren
db:drop               Drops the database from DATABASE_URL or config/database.yml for the current
db:fixtures:load                          Load fixtures into the current environment's da
db:migrate                  Migrate the database (options: VERSION=x, VERBOSE=false, SCOPE
db:migrate:status                                          Display status of migi
db:rollback                  Rolls the schema back to the previous version (specify steps w/ ST
db:schema:cache:clear                                          Clear a db/schema_cache.du
db:schema:cache:dump                                          Create a db/schema_cache.du
db:schema:dump           Create a db/schema.rb file that is portable against any DB supported
db:schema:load                                          Load a schema.rb file into the da
```

```
def destroy
  @article = Article.find(params[:id])
  @article.destroy

  redirect_to articles_path
end

private

def article_params
  params.require(:article).permit(:title, :text)
end
```

Figure 7:RubyMine interface

j) Results :

For the moment we have the authentication and the home page, where you are able to sign up using your email, and make your own account on the website, and use it to log in after confirmation by clicking on the link that you receive automatically after signing up, then you can use the compiler to compile your own code, moreover you can save the code that you have written and it would be associated to your account,
and the results of the web that we have achieved until the moment looks like the following images :

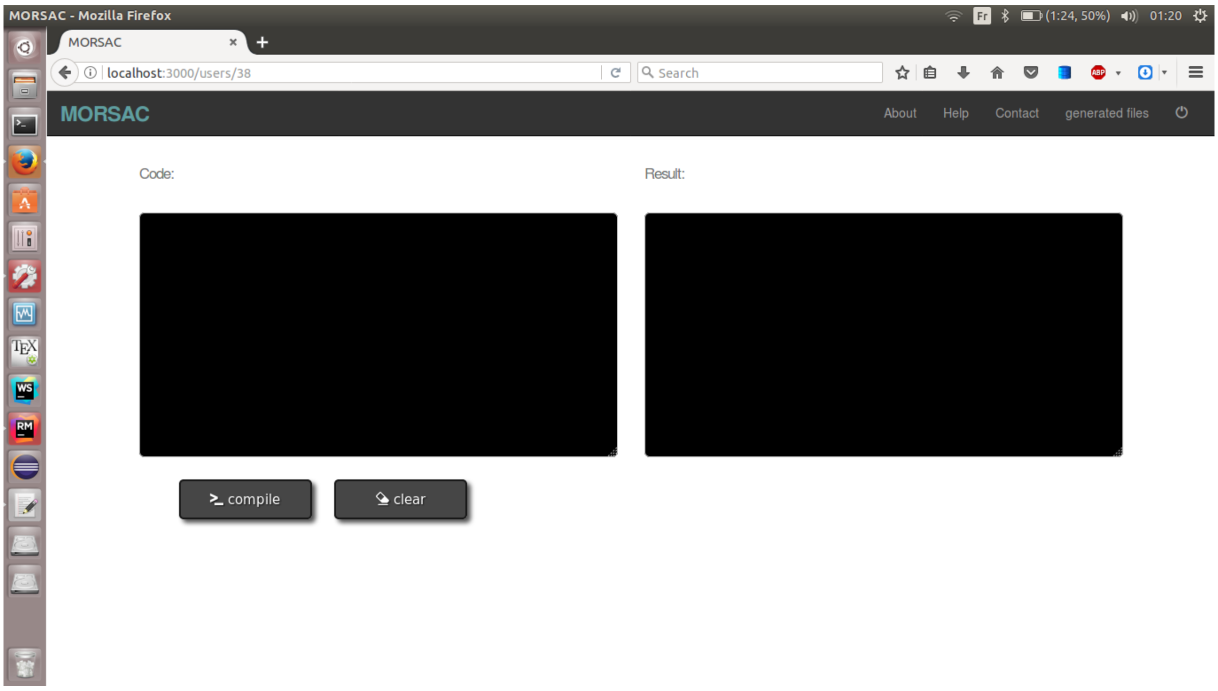
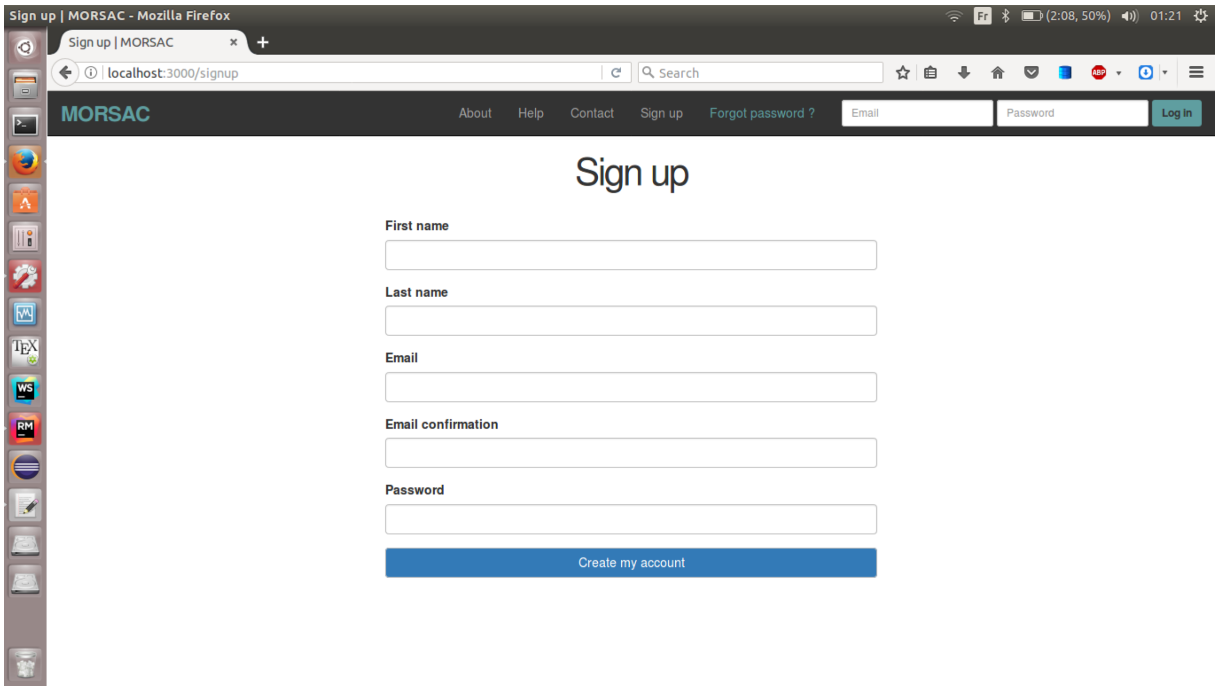


Figure 8:Home Page

This figure represents the most important part of the website, it actually illustrates the main aim of our project. In this screen-shot are two fields of text and two buttons, the text field labeled 'code' is for typing a source code to be compiled by the server using the compile button and a second button to clear the screen, on the right side is the result output that displays either a message telling that the compilation went well and therefore the corresponding program has been generated in the user space, or a compilation error message would be generated which would help the user to debug the source code.



Sign up

First name

Last name

Email

Email confirmation

Password

Create my account

Figure 9:Sign_up page

This figure illustrates our working sing up form to allow the creation of new users, along with a header that contains a form to let users log into the site and a password reset link to recover lost passwords. A visitor can register by using the HTML form with all the required registration fields. Now, after submitting the user information to be saved to the database, a user must activate his account through email in order to interact on the site. The email activation process verifies and confirms that he has given his real email address. For security purposes, the user password is hashed with an irreversible hash function then stored in database, this means even if our database is compromised, our users passwords will still be secure. Once the user logged in, we are setting a persistent session by creating cookies which are small pieces of text placed on the user's browser, the drawback of persistent cookies is that they are vulnerable to be intercepted and used by a hacker to log in with someone else's account, that's why we are storing a hash of the cookie instead of the cookie itself in much the same way that we did with the password, furthermore we are changing the cookie every time a user logs out, and this is important to prevent someone who has gained a physical access to a machine with logged in user.

5) Project deployment :

k) what to know before using the cloud :

i) Definition :

Cloud computing can be defined in too many different ways, but generally is a general term for the delivery of hosted services over the internet, specifically it enables companies to consume a compute resource, such as a virtual machine or an application, rather than having to build and maintain computing infrastructures in your company or even in your house.

it can be represented as it is shown in the following image:

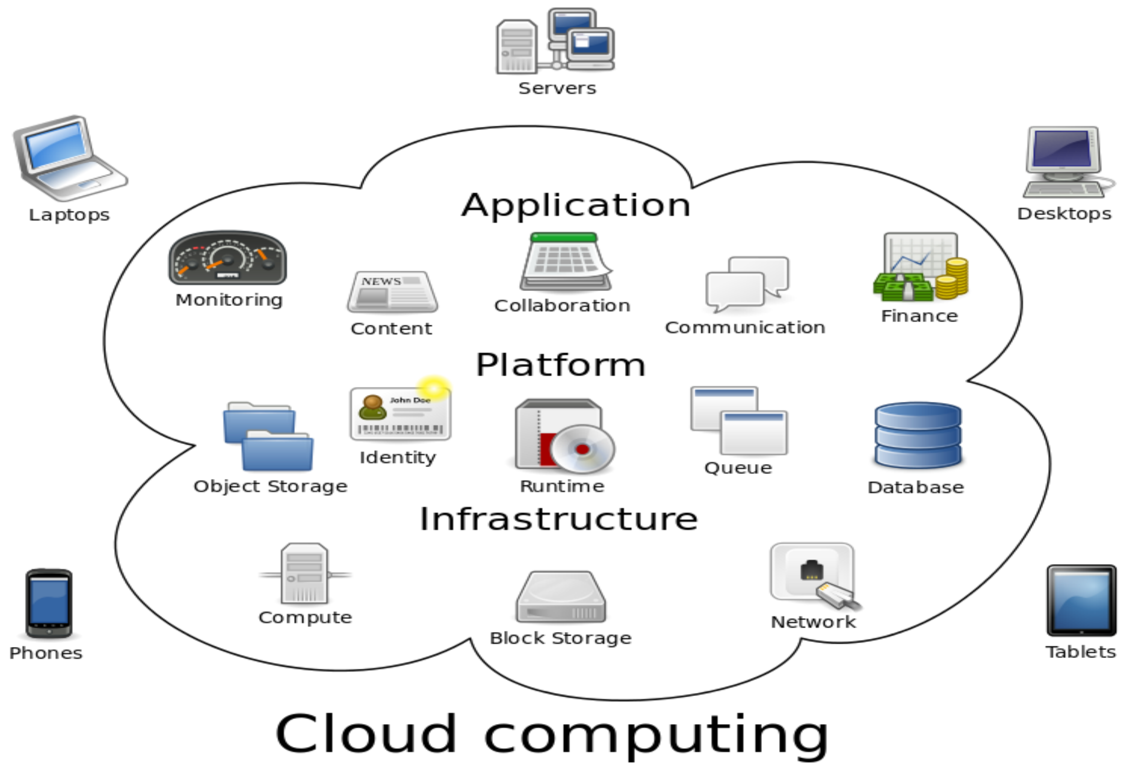


Figure 10: Cloud computing

And it is not only about virtualization, the cloud ensures you the efficiency, elasticity and scalability and doing things faster and better.

ii) Cloud models :

The cloud can be deployed through many models like:

- ✓ Infrastructure as a service (IaaS): is the delivery of components such as hardware, software, storage or even networking.
- ✓ Software as a service (SaaS) : is the most familiar form of cloud service for consumers , and it is where clients are usually web browsers, provide the point of access to software running on servers.
- ✓ Platform as a service (PaaS) : is a combination of IaaS and set of middleware, and it functions at a lower lever than SaaS , providing a platform on which software can be developed and deployed .

  the following image explains these three models of cloud (web hosting) :

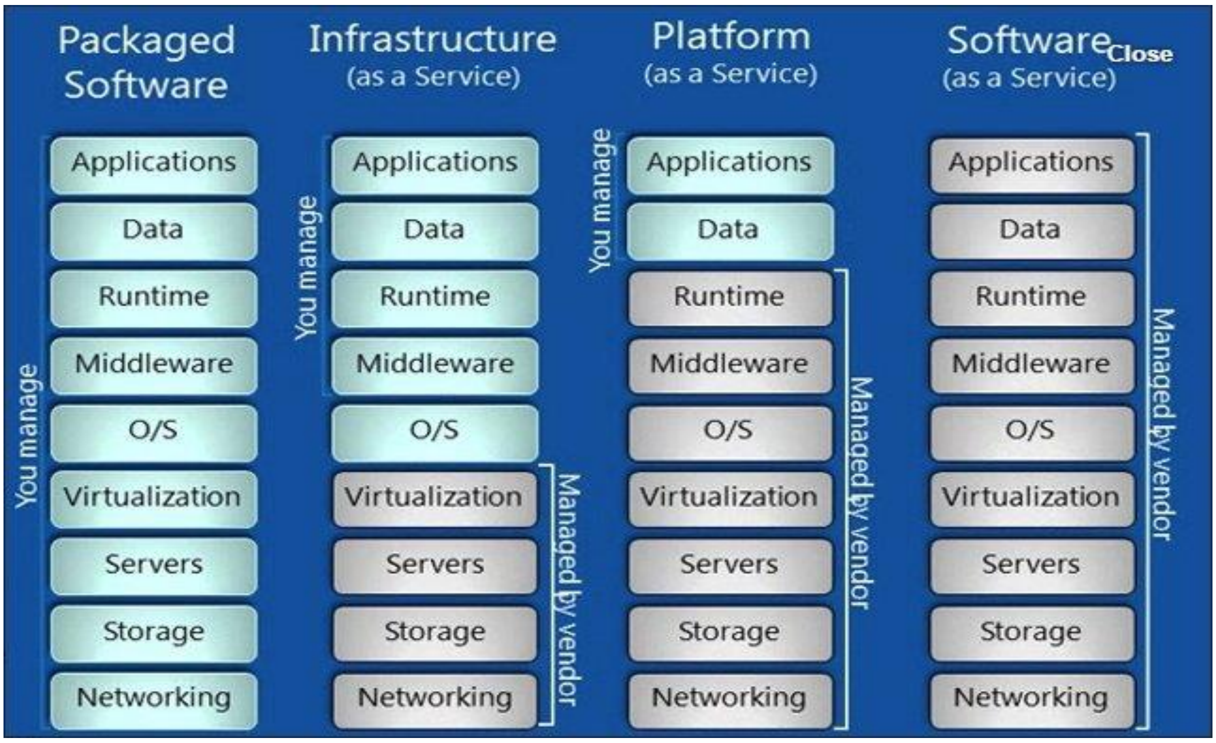


Figure 11: Cloud models

iii) Transformation to cloud :

Before moving to a cloud architecture and for a successful cloud adoption, you need to understand your IT environment by identifying the workloads that will fit your cloud target, therefore, you should proceed by the following verification and testing for your environment :

- **Business impact** : you should start with an application that has the less impact for your business, to get more experience before moving that workload.

- **Production versus development and test** : the golden rule always to start with less critical development and test environment first, and considering your production.

- **Performance :** you should take it into consideration more than any other arguments in environments, such as the extensive data processing ; because what matters most for users is the response time with applications used.

- **Complexity :** you should avoid complex architecture systems with several points of integration between applications.

- **Licensing :** analyze your software vendor's cloud model and see if it would be costly to change from a dedicated model.

- **Security :** validate if the cloud vendor supports security constraints required from your business.

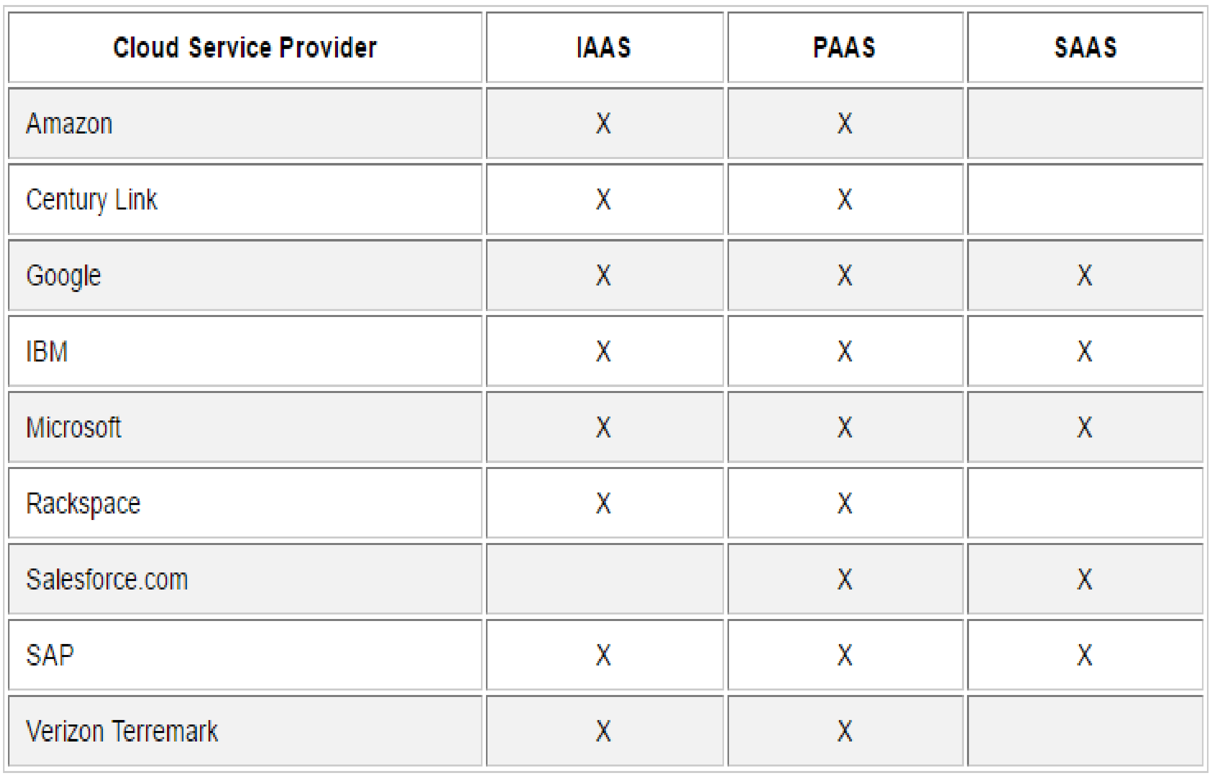
- **Platform :** verify the operating system version required for your application before choosing the platform.

- **Data Hosting :** make sure that your vendor has a cloud data center in your country in the case where you are not allowed to host some critical data outside.

- **Customize your application :** in order to fit the cloud requirements, you need to transform your application into a web-based application if it is not the case.

iv) Choosing the vendor :

This step is a relative one, depends on your needs. therefore this table below shows a list of some of the major type cloud service provider and their approaches:



Cloud Service Provider	IAAS	PAAS	SAAS
Amazon	X	X	
Century Link	X	X	
Google	X	X	X
IBM	X	X	X
Microsoft	X	X	X
Rackspace	X	X	
Salesforce.com		X	X
SAP	X	X	X
Verizon Terremark	X	X	

Figure 12 : cloud providers

Finally, the cloud is a choice that can simplifies life and economizes costs for companies but you should always take into consideration the point required before moving to this approach.

6) Conclusion :

In a nutshell, this project has been a great rewarding experience for both soft and technical skills, MORCSAC introduced us to the ruby programming language and helped us to become familiar with the cloud computing concept which nowadays, has become a great solution for providing a flexible, on-demand, and dynamically scalable computing infrastructure for many web-based applications.

And in terms of human relationships, we have very much benefited from the agile development methodology which is achieved by establishing regular cadences of work, known as sprints or iterations, of each member of our project.

We had faced some difficulties because we hadn't no experience in Ruby on Rails whatsoever, however with the time spent on research and documentation we improved our knowledge in ROR the most popular and perfect framework for designing web applications which may helps us in our internship or in our future work.

Glossary

Abbreviation	Definition
IDE	Integrated development environment
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
XML	Extensible Markup Language
URL	Uniform Resource Locator
ROR	Ruby On Rails
MORCSAC	Méthodes, Outils et Règles de Conception pour la mise en place de services SaaS dans le Cloud: cas des compilateurs en ligne
IT	Information Technology
CSS	Cascading Style Sheets

Bibliography

(s.d.). Récupéré sur railstutorial: https://www.railstutorial.org/

Bigg, R. (2015). *Rails 4 in action.*

IBM. (s.d.). *IBM*. Récupéré sur Cloud IBM: https://www.ibm.com/blogs/cloud-computing/

KORCHI&KARKOUB. (2017). *Github/MORCSAC*. Retrieved from site web Github: https://github.com/MORCSAC

sdxcentral. (s.d.). Récupéré sur site web . sdxccentral : https://www.sdxcentral.com/cloud/definitions/what-are-cloud-service-providers/

Starting Ruby On Rails :What I Wish I Knew. (s.d.). Récupéré sur betterexplained: https://betterexplained.com/articles/starting-ruby-on-rails-what-i-wish-i-knew/

Wiki. (s.d.). *Wikipedia/Cloud*. Récupéré sur Wikipedia: Wikipedia.org

l) Appendix A: Status report

Status Report

MORCSAC

Summary

I. Introduction :.. **Error! Bookmark not defined.**

II. The project context : 24

1. Definition and objective : 24
2. Web Technologies : 24

A. The client side:.. 24

B. The server side:... 25

3. Prototype and language : 25

III. Conclusion :.. 26

(1) Introduction:

In order to follow the SaaS (Software as a service) approach, which simplifies the life of users and developers (no need to install software to use it), we were asked to make a website where you would be able to use a compiler (represents the software) developed in our school as an online service.

This report explains the technologies and tools that are available and exploitable to develop the website, also the method used in order to dispatch the tasks.

(2) The project context:

(a) Definition and objective:

The project goal is defining and building a website that allows compiling a code and getting the results. In other words, it transforms the source code written in a certain programming language (the source language) into another computer language and creates an executable program then sees the results of your program.

(b) Web Technologies:

In terms of technologies that can be used to proceed in the project, we find plenty of choices; and they can be classified as technologies for the client side and the server side, since that website is split up to these latter sides:

(i) The client side:

Firstly, we can describe this side as the code that lives in the browser and responds to the user input.

For languages that are used we find:

- HTML : which is the standard markup language to start creating web pages.

- CSS : is a cascading Style Sheet. It allows web designers to change colors, animations and transitions on the web and to smarten up the web.

- JavaScript : a programming language that makes the website pages dynamic and interactive.

(ii) The server side:

It is the code that lives on the server and responds to HTTP requests ; and this side cannot be seen by the user.

Since any code that can be executed on a computer and respond to HTTP requests can also run a server, then we are in front of a lot of frameworks and languages to develop on this side, such as:

- PHP : is an HTML-embedded web scripting language, which means that it can be inserted into the HTML of a web page.

- Django : which is a free web framework, written in PYTHON (programming language) and its architecture pattern is based on the standard of MVT (model-view-template).

- Rails :is a MVC (model-view-controller) web framework that is written in RUBY. Its particularity lies in providing default structures of a database, a web service also web pages.

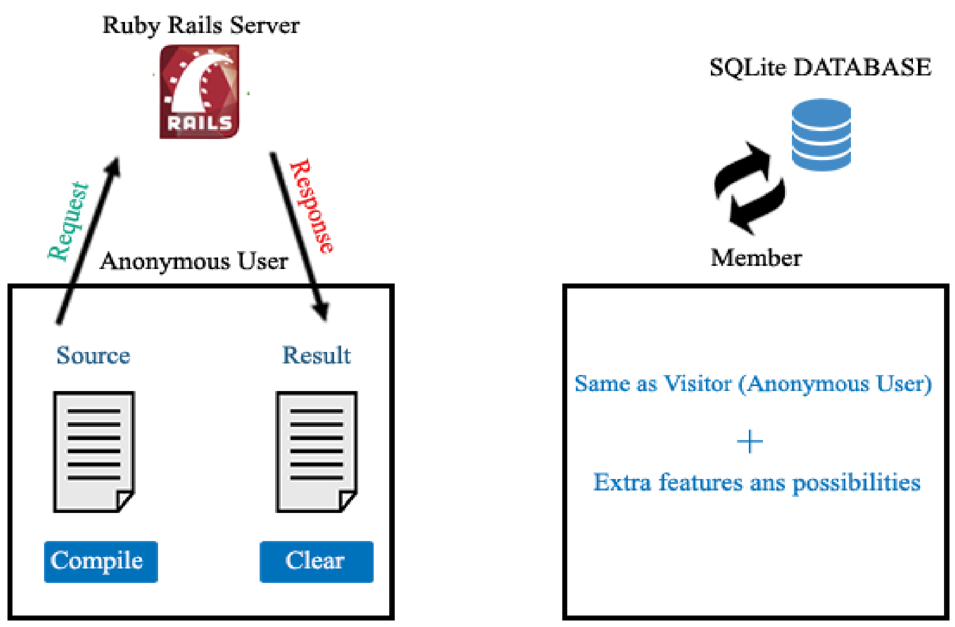
- C# :is an object-oriented programming language ,which is marked by its rapidity of compiling.

(c)Prototype and language:

The particularity of our project lies in the fact that our supervisor gave us the voice and choice in terms of web technologies and features that the website can have in order to take responsibility and to fit our own style when it comes to web programming. Needless to say, we have to meet some requirements such as using Ruby as a programming language for the application.

After a thirty-minute meeting with our supervisor, he instructed me and my colleague how we are going to proceed to build an application prototype as a first step and also to brainstorm great ideas that we are going to shape during the project.

Our prototype is illustrated as follows:



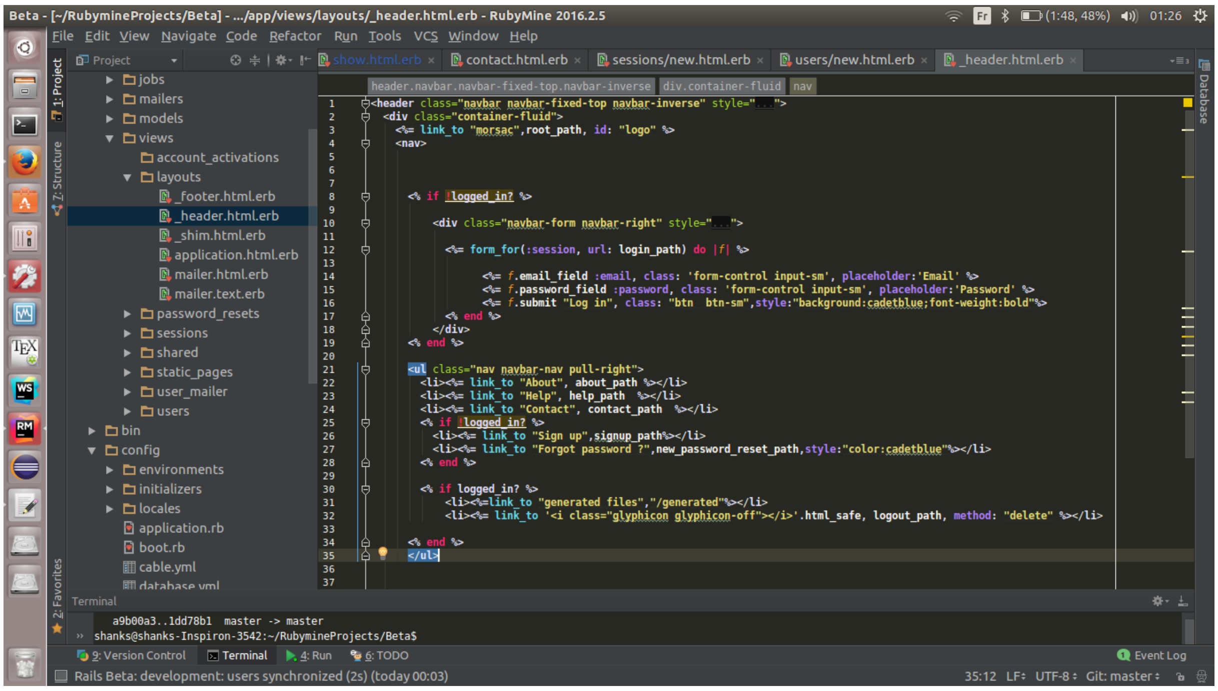
As we agreed in our last meeting, there will be two kinds of interactions with the website, the first one is as visitor (anonymous user) to test the application and to see what is it about. Furthermore, any user can register and become a member. In terms of web design, the concept works for both cases (visitor and member), however the logged-in user can have access to a range of features with more possibilities that we still don't define for the moment.

As everybody knows, HTTP is the protocol used by browsers (client side) and the web server, it is based on request/response: once a user fills the source text field with his/her code and hits Compile, the browser makes an HTTP POST in order to send the code to be compiled by the server. After processing the request, the server returns the results and writes them in the Result text field to see whether the compilation went ok or not.

(3) Conclusion :

To conclude, the application prototype is progressing well, it has been held up slightly waiting for documentation and a great deal of knowledge has been gained through this phase. At the moment, all needed parts have been obtained for the prototype improvement. As we stated before, the project has no limit in terms of features and services that the website offers.

m) Appendix B : Login code

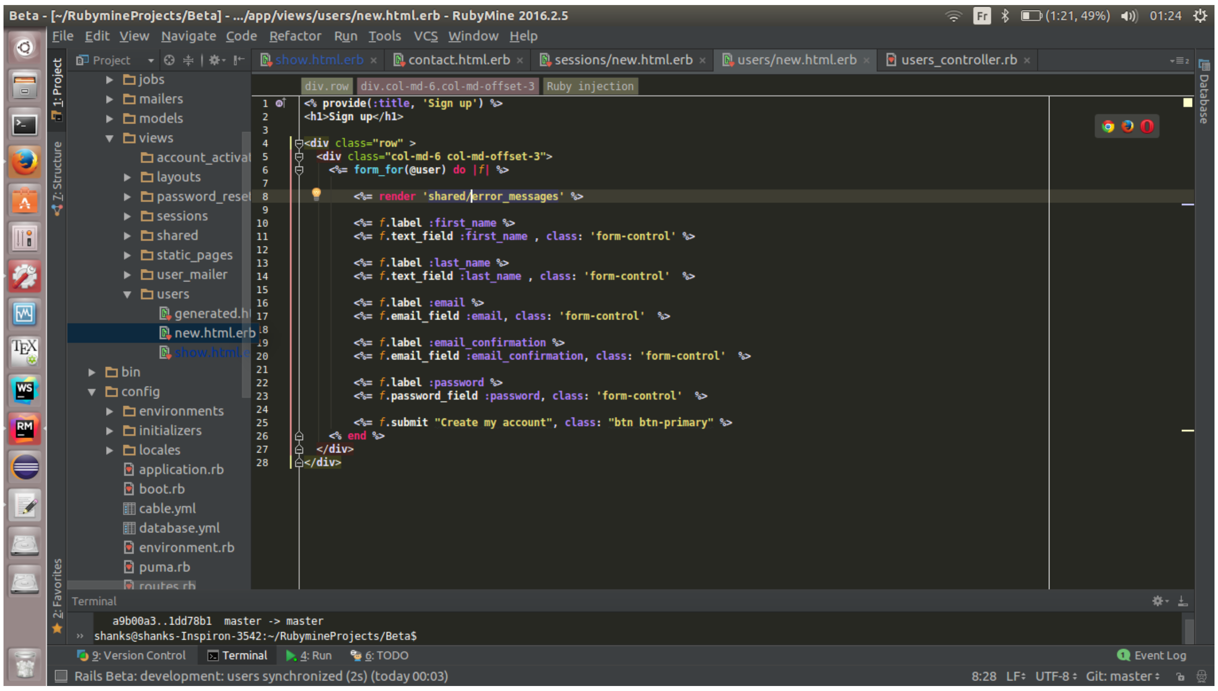

Beta - [~/RubymineProjects/Beta] - .../app/views/layouts/_header.html.erb - RubyMine 2016.2.5
File Edit View Navigate Code Refactor Run Tools VCS Window Help
show.html.erb
contact.html.erb
sessions/new.html.erb
users/new.html.erb
_header.html.erb
header.navbar.navbar-fixed-top.navbar-inverse
div.container-fluid
nav
Project
Terminal
9: Version Control
Terminal
4: Run
6: TODO
Event Log
Rails Beta: development: users synchronized (2s) (today 00:03)
35:12 LF÷ UTF-8÷ Git: master÷

```
<header class="navbar navbar-fixed-top navbar-inverse" style="...">
  <div class="container-fluid">
    <%= link_to "morsac",root_path, id: "logo" %>
    <nav>



      <% if !logged_in? %>

          <div class="navbar-form navbar-right" style="...">

            <%= form_for(:session, url: login_path) do |f| %>

                  <%= f.email_field :email, class: 'form-control input-sm', placeholder:'Email' %>
                  <%= f.password_field :password, class: 'form-control input-sm', placeholder:'Password' %>
                  <%= f.submit "Log in", class: "btn  btn-sm",style:"background:cadetblue;font-weight:bold"%>
            <% end %>
          </div>
      <% end %>

      <ul class="nav navbar-nav pull-right">
        <li><%= link_to "About", about_path %></li>
        <li><%= link_to "Help", help_path  %></li>
        <li><%= link_to "Contact", contact_path  %></li>
        <% if !logged_in? %>
          <li><%= link_to "Sign up",signup_path%></li>
          <li><%= link_to "Forgot password ?",new_password_reset_path,style:"color:cadetblue"%></li>
        <% end %>

        <% if logged_in? %>
            <li><%=link_to "generated files","/generated"%></li>
            <li><%= link_to '<i class="glyphicon glyphicon-off"></i>'.html_safe, logout_path, method: "delete" %></li>

      <% end %>
      </ul>
```

```
   a9b00a3..1dd78b1  master -> master
shanks@shanks-Inspiron-3542:~/RubymineProjects/Beta$
```

Figure 13:Login Code

n) Appendix C : signup code



```
Beta - [~/RubymineProjects/Beta] - .../app/views/users/new.html.erb - RubyMine 2016.2.5
```

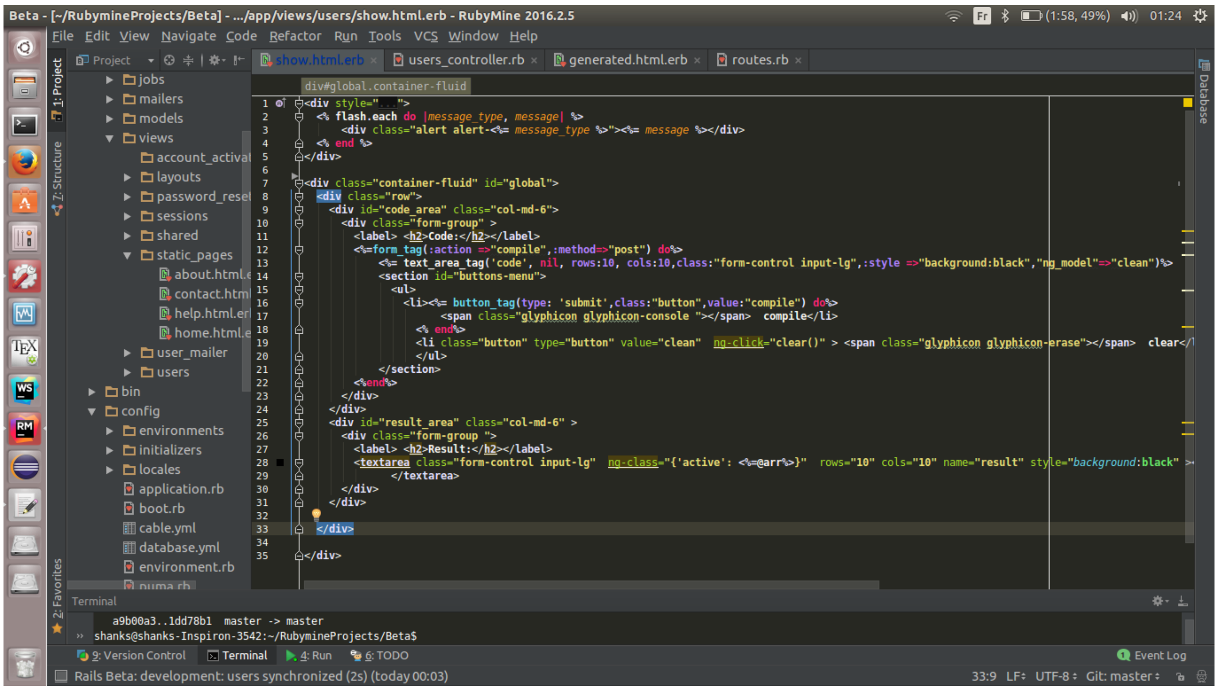
```erb
<% provide(:title, 'Sign up') %>
<h1>Sign up</h1>

<div class="row" >
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(@user) do |f| %>

      <%= render 'shared/error_messages' %>

      <%= f.label :first_name %>
      <%= f.text_field :first_name , class: 'form-control' %>

      <%= f.label :last_name %>
      <%= f.text_field :last_name , class: 'form-control'  %>

      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control'  %>

      <%= f.label :email_confirmation %>
      <%= f.email_field :email_confirmation, class: 'form-control'  %>

      <%= f.label :password %>
      <%= f.password_field :password, class: 'form-control'  %>

      <%= f.submit "Create my account", class: "btn btn-primary" %>
    <% end %>
  </div>
</div>
```

```
a9b00a3..1dd78b1  master -> master
shanks@shanks-Inspiron-3542:~/RubymineProjects/Beta$
```

```
Rails Beta: development: users synchronized (2s) (today 00:03)        8:28  LF÷  UTF-8÷  Git: master÷
```

Figure 14:sign up code

o) Appendix D: Home code



```
Beta - [~/RubymineProjects/Beta] - .../app/views/users/show.html.erb - RubyMine 2016.2.5
```

```erb
<div style="...">
  <% flash.each do |message_type, message| %>
      <div class="alert alert-<%= message_type %>"><%= message %></div>
  <% end %>
</div>

<div class="container-fluid" id="global">
  <div class="row">
    <div id="code_area" class="col-md-6">
      <div class="form-group" >
        <label> <h2>Code:</h2></label>
        <%=form_tag(:action =>"compile",:method=>"post") do%>
            <%= text_area_tag('code', nil, rows:10, cols:10,class:"form-control input-lg",:style =>"background:black","ng_model"=>"clean")%>
            <section id="buttons-menu">
              <ul>
                <li><%= button_tag(type: 'submit',class:"button",value:"compile") do%>
                      <span class="glyphicon glyphicon-console "></span>  compile</li>
                  <% end%>
                  <li class="button" type="button" value="clean"  ng-click="clear()" > <span class="glyphicon glyphicon-erase"></span>  clear</l
                  </ul>
            </section>
        <%end%>
      </div>
    </div>
    <div id="result_area" class="col-md-6" >
      <div class="form-group ">
        <label> <h2>Result:</h2></label>
        <textarea class="form-control input-lg"  ng-class="{'active': <%=@arr%>}"  rows="10" cols="10" name="result" style="background:black" >
            </textarea>
      </div>
    </div>

  </div>

</div>
```

```
a9b00a3..1dd78b1  master -> master
shanks@shanks-Inspiron-3542:~/RubymineProjects/Beta$
```

```
Rails Beta: development: users synchronized (2s) (today 00:03)        33:9  LF÷  UTF-8÷  Git: master÷
```

Figure 15: home code