MILESTONE 3

PROJECT WEB

GRP: A | RAHAMIM SITBON
KEENAN BARANES
NOE BRUMELOT

PURPOSE

The objective of this project is to create a To Do List in the form of a website. The concept of a “To Do List” is a list in which you can add tasks that you have to do and to cross if they’re either started, in progress or finished. On our website, every single person will be able to create an account on which he’ll have access to his own list and on which every modification he does is saved. So we have to use a database to save each list with the corresponding user.

We decided to name our project “Just Do It” as we’ll try to fight procrastination with it. It’ll be composed of at least 4 pages, one for logging in, one for signing up, the main one which is the To Do list itself, aswell as a contact page. If we have the time we’d like to add a home page and maybe another one listing every member of the Just Do It community.

STRUCTURE

FRONT-END

The front end is what the user sees while using the website and the interaction he has with it, so for these parts we decided to use vue.js which is the third most popular application for front-end in term of framework in the world.

BACK-END

Node.js is a free software platform in JavaScript, suitable for network applications.

DATABASE

To create our database, we will be using the MySQL application and we are going to link it to our website. MySQL is a relational database management system. It is distributed under a dual GPL and has a proprietary license.

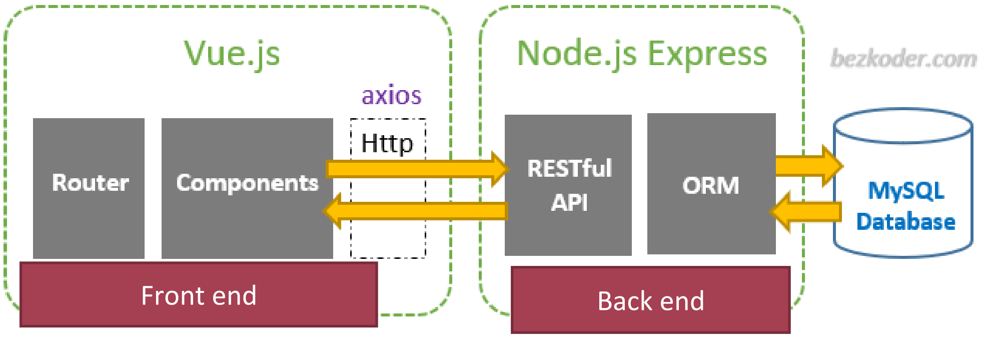
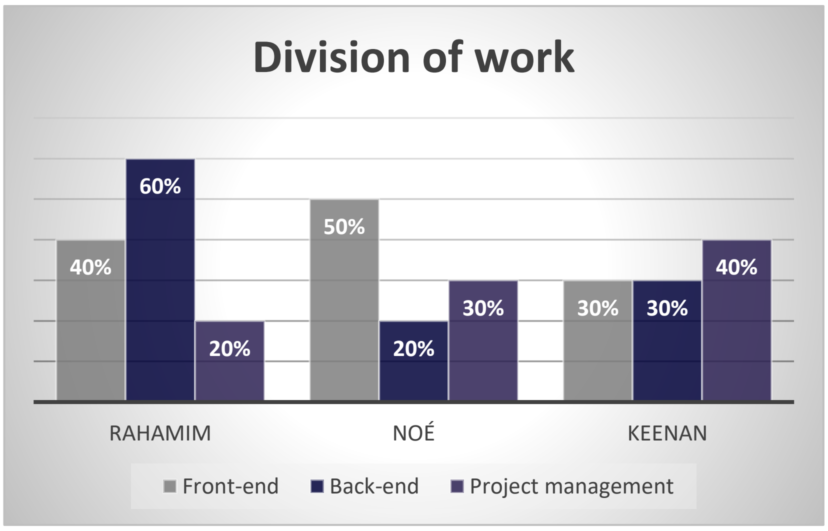


Figure 1: Architecture of our TODO LIST

WORK DIVISION



	Front-end	Back-end	Project management
RAHAMIM	40%	60%	20%
NOÉ	50%	20%	30%
KEENAN	30%	30%	40%

DIAGRAMS & METHODS

METHODE USED

- KABAN
- SOLID
- DRY

PRIORITY

- Use modern programming practices.
- Design first, then build.

ABOUT US

- Group of 3
- 3 parts of projects

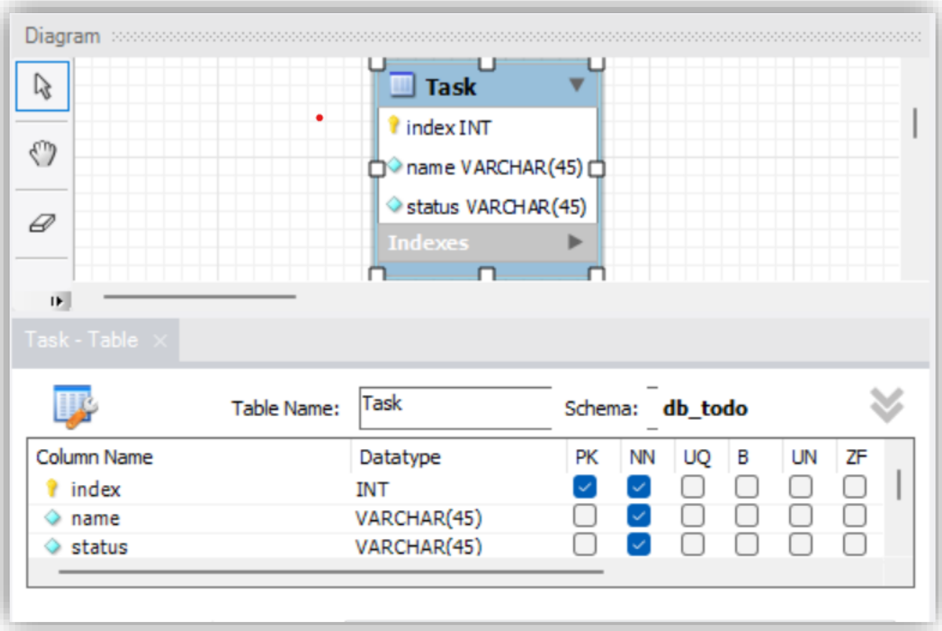


Table Name: Task

Schema: db_todo

Column Name	Datatype	PK	NN	UQ	B	UN	ZF
index	INT	✓	✓	☐	☐	☐	☐
name	VARCHAR(45)	☐	✓	☐	☐	☐	☐
status	VARCHAR(45)	☐	✓	☐	☐	☐	☐

CREATE TABLE

Thanks to the UML diagram, we get that SQL code:

As a result, we can initialize our tables for the database.

```sql
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema db_todo
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `db_todo` DEFAULT CHARACTER SET utf8 ;
USE `db_todo` ;

-- -----------------------------------------------------
-- Table `db_todo`.`Task`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_todo`.`Task` (
  `index` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `status` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`index`))
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

COMMUNICATION UML DIAGRAM

A communication diagram is a UML 2.0 interaction diagram, a simplified representation of a sequence diagram focusing on message exchanges between objects

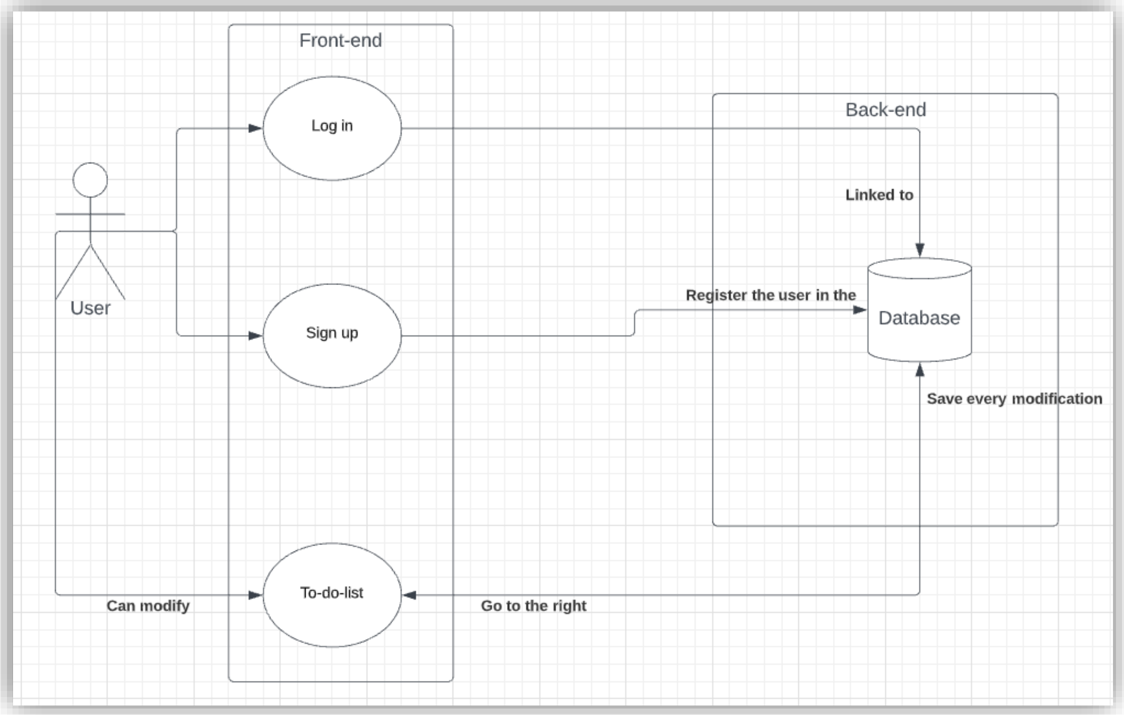
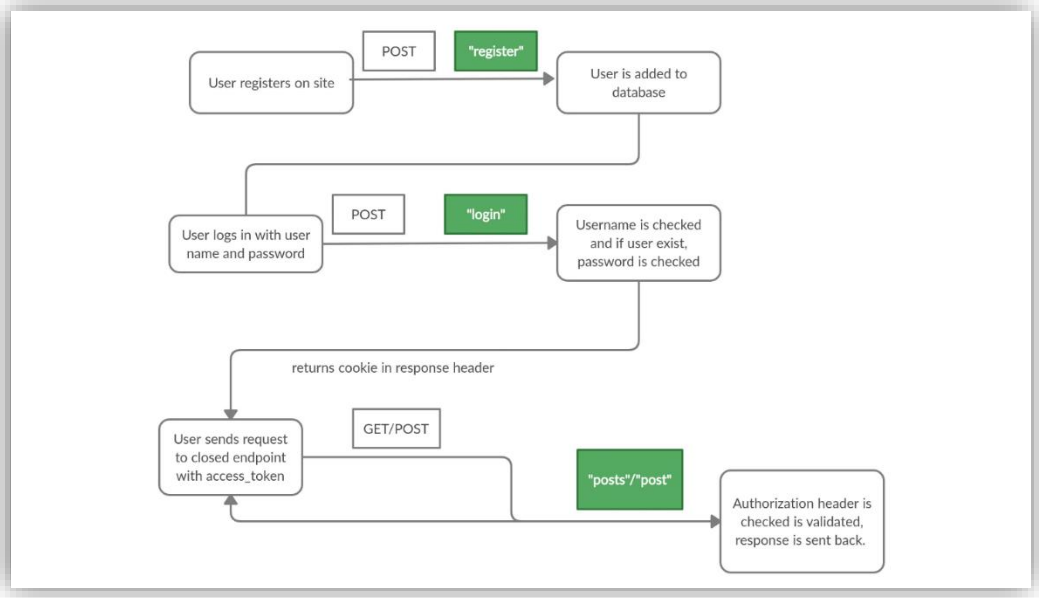


Figure 2: UML Diagram communication

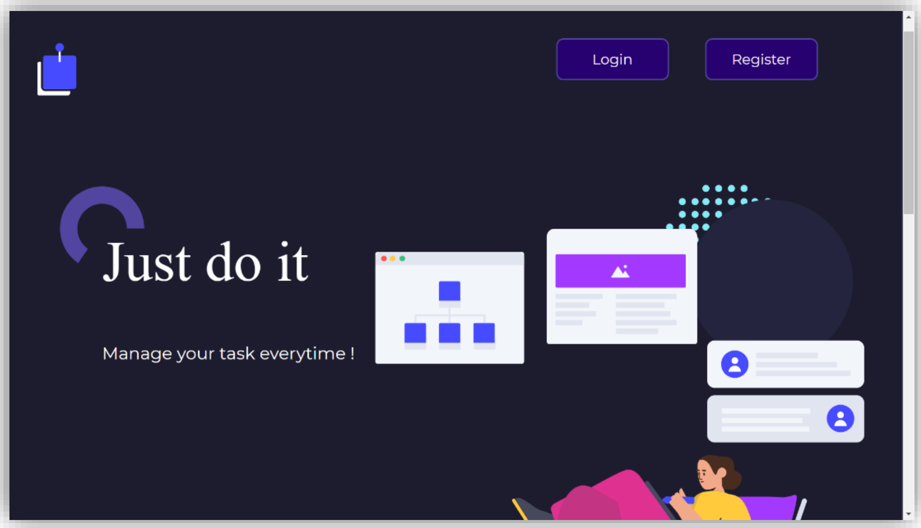
BUSINESS LAYER


POST
"register"
User registers on site
User is added to database
POST
"login"
User logs in with user name and password
Username is checked and if user exist, password is checked
returns cookie in response header
GET/POST
User sends request to closed endpoint with access_token
"posts"/"post"
Authorization header is checked is validated, response is sent back.

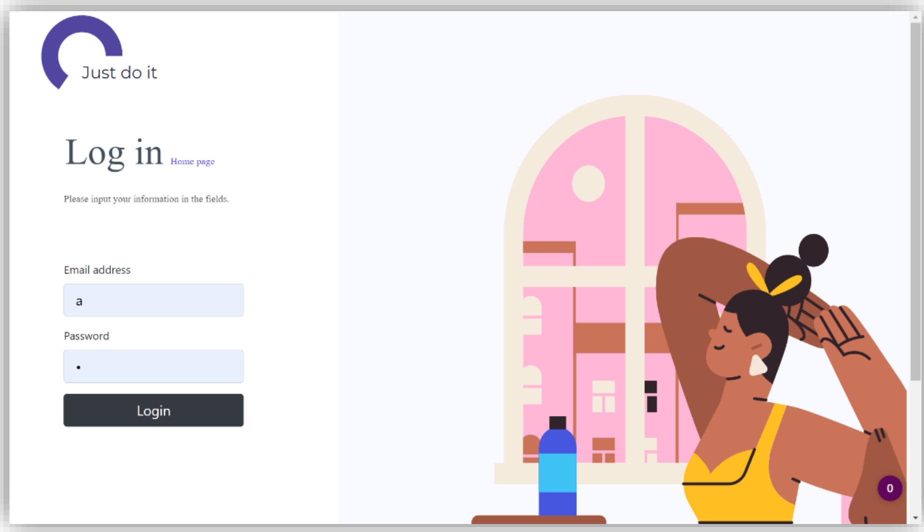
WEBSITE PAGES

Finally, we managed to have four different pages in our web site:

- Home page: where you can go either on login page our register page thanks to buttons.



- Login page: you can enter your mail address and password to access you list

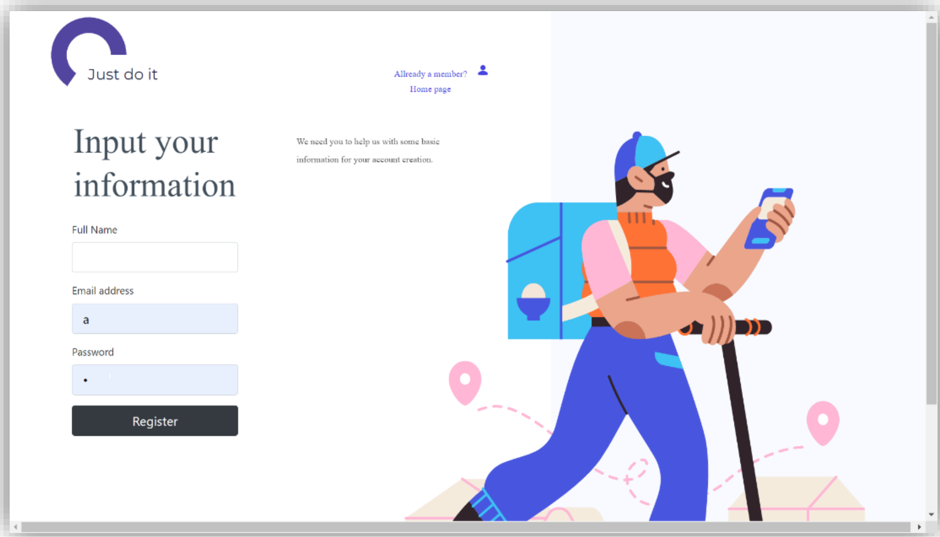


Please input your information in the fields.

Email address: a

Password: •

- Register page: you can create an account to have a blank list



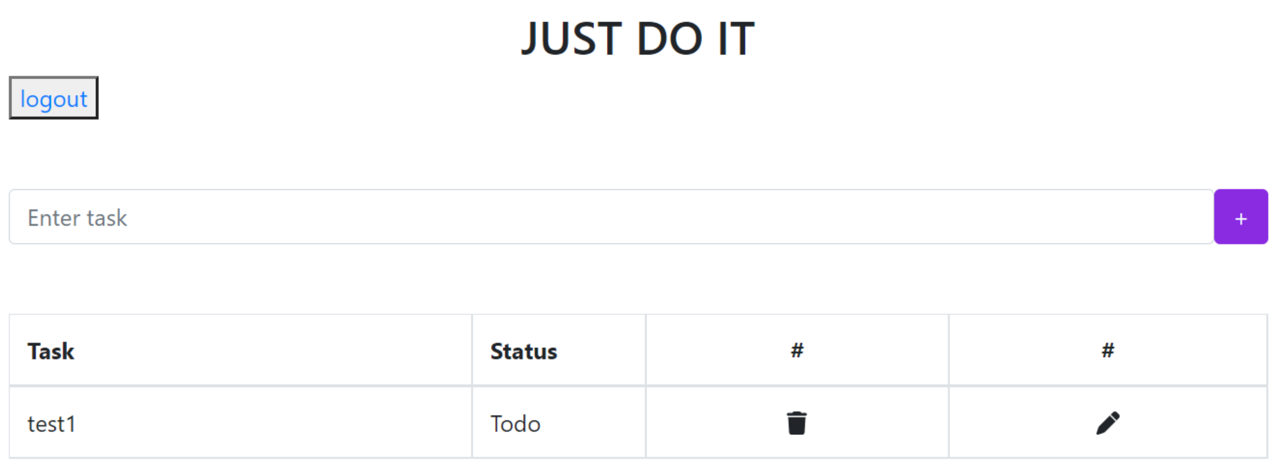
We need you to help us with some basic information for your account creation.

Email address: a

Password: •

- To-do-list page: You can add and delete tasks from your list, change their status (to do, doing or finished)

Example of a to-do-list on our web site



JUST DO IT

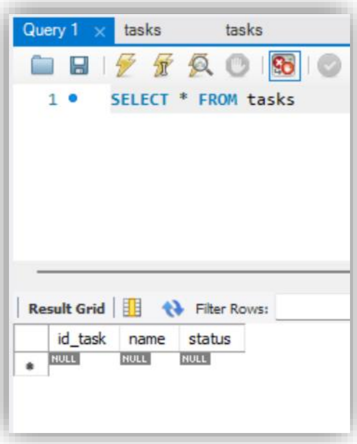
logout

Enter task +

Task	Status	#	#
test1	Todo	🗑	✎

N O D E. J S & M y S Q L

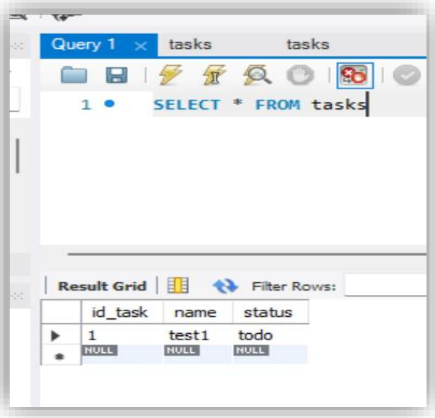
Thanks to Node.js, we have linked our web site to a database that will get every task you add in your list. If you open MySQL after adding a task in your list, you will see it in your database table.



```
1  SELECT * FROM tasks
```

id_task	name	status
NULL	NULL	NULL

After add task « test1 »



```
1  SELECT * FROM tasks
```

id_task	name	status
1	test1	todo
NULL	NULL	NULL

As we see in those screenshots, the task as been added successfully in the database. To make this possible, firstly, we need to open MySQL to make the database accessible. Then we launch the server on the folder where we have our data for the web site thanks to "nodemon index.js". For the client we use the command "run serve".

P O S S I B L E I M P R O V E M E N T

We didn't have the time to have the perfect web site as we wanted it to be. So we have some ideas about how it could have been improved with a little bit more time. Since the database is linked to the To-do-list, every task added is send to the database, but we didn’t have the time to add some function. For example when you modify the name of a task, the databse won’t save it, and it’s the same if you delete a task. As a result, we would have liked to add the modifying and deleting functions to our program.

Table des matières

R E F E R E N C E S

https://www.youtube.com/watch?v=X__rLNfTsLg&t=75s

https://www.youtube.com/watch?v=YfaLLVJwyes&t=31s

https://www.youtube.com/watch?v=xZMwg5z5VGk

https://www.bezkoder.com/serve-vue-app-express/

https://www.bezkoder.com/vue-js-node-js-express-mysql-crud-example/