



CODING A SCHOOL SCHEDULE

SERGONNE LOU

TRAVAIL PERSONNEL 2022-23

TUTEUR: ALEX BARA

Introduction

This year, I have taken on a personal project to develop a web application and establish the essential infrastructure to support it. The purpose of this project is to create a web application that assigns students to the Enterprises in the LEM. Basically, the web app enables Enterprise staff to see if students are present or expected, locate them if they're absent, and provide easy access to student information like IAM, email, full name, and other functions.

Having started coding since 6ième, I have gained some experience in the field of coding and started getting passionate about it. Initially, I taught myself Python during my free time, and my first project was to build a password manager. Though I faced some complications with encryption, hashes, and salts, I am still working on it. Furthermore, I have developed several websites and scripts to make the work in the Enterprise and in everyday life in general more manageable.

The inspiration for this project emerged during a conversation with my Enterprise chef and Tuteur, Alex Bara. We were wondering, if there was a better way to manage student attendances since the current method to manage it is chaotic and messy. It is mainly consisting of two Excel sheets and one Word document, and we wanted to improve that.

As I began developing the concept and designing the web app, I realized that this project would be ideal for my personal project of this year. This project gives me the chance to evolve in the field I aim to work in someday, and it offers an enjoyable opportunity to learn about the involved processes.

Next to that, this project has the potential to benefit not only my learning but gives me the opportunity to analyse and to understand this specific eco-system and so to simplify the management of the Enterprises and the students involved. By streamlining the process of managing students and providing easy access to their credentials, a more efficient and organized system might be established in the future.

I really hope you enjoy reading about my personal project and find it interesting and fun. I've put a lot of heart into it!

Table of contents

Introduction	3
Table of contents	4
Plan	6
Phase 1: Wireframing	7
<i>First Versions of the Designs</i>	8
<i>Versions afterwards</i>	12
<i>Final Version</i>	16
Coloured version	17
Phase 2: Database Setup	19
<i>Database client</i>	19
<i>First encounter with a Database</i>	22
The basics	23
Basic Structure of a Database	24
<i>Second try of a database</i>	25
<i>Optimising the database</i>	28
<i>Mini automation project for PHP structure</i>	29
Phase 3: Coding	34
<i>GitHub</i>	34
Folder Structure	34
<i>Dimensions of the repository</i>	35
<i>Overview of Folder Contents</i>	37
Scripts	37
What are those scripts?	37
Contents of the folder	37
CSS	37
What is CSS?	37
Folder contents	37
Database	38
<i>Insight and explanation of a piece of code</i>	38
<i>Procedure of coding</i>	43
<i>GitHub Changelog</i>	44
The 5 first commits	44
The 10 last commits	44
<i>Test Hosting</i>	45
Sftp	45
Phase 4: Import of Data	46
<i>Inserting real life data</i>	47
Phase 5; Testing	48

<i>Steps of testing</i>	48
Functional testing	48
Database testing	48
Performance testing	48
Security testing	48
Usability testing	48
Compatibility testing	48
Phase 6: Deployment	49
Problems while working	49
<i>General Difficulties</i>	49
Time constraints.	49
Working time calculations	49
Time constraint	51
<i>Learning part</i>	51
Definitions	53
<i>Design</i>	53
Wireframes	53
Figma	54
Colors	55
<i>Databases</i>	56
Databases in General	56
SQL	56
MySQL	56
PhpMyAdmin	57
Foreign keys	57
Primary keys	57
Unique fields	57
Indexing tables	58
One to one, many to many, etc	58
Mockaroo	59
<i>Coding</i>	60
Python	60
Web	60
HTML	60
JavaScript	61
PHP	61
CSS	62
Git	62
GitHub	63
VSCode	63
Pastebin	63
GUI	63
Conclusion	64
Source and Inspiration	66
<i>Learning Resources</i>	66
<i>Information resources</i>	66
<i>Human resources</i>	66

Plan

This year, I have a plan for my personal project that I have divided into several steps. To ensure proper organization, I sought advice from a web design company called Nvision and tried to replicate their project creation process.

The plan for my TraPe is divided in following steps:

1. Wireframe and Feature planning
 - a. Plan the website in Figma.
 - b. Colour scheme
 - c. Feature listing
2. Database Structure and test Data
 - a. Choose Database type and Client.
 - b. Database Structure to build my website off.
 - c. Import random generated data for testing purposes.
3. Coding
 - a. Decide which Languages to use.
 - b. Writing the code
 - c. Testing the Code
4. Importing Data
 - a. Import Real-life data to find further bugs and make improvements.
 - b. Test the database efficiency.
5. Testing
 - a. Test the Website or let users test it.
 - b. Test the security.
6. Deployment
 - a. Deploy the website to a domain and host it on a server.

This is the plan I worked out in the beginning to create a more structured project.

Phase 1: Wireframing

Building a wireframe in Figma is an effective way to begin planning out the structure and layout of your website or application. By creating a basic framework or skeleton, you can easily make changes and adjustments to the design without committing to a final version.

As I found out while, building a wireframe, it is important to focus on the functionality and user experience of the product, rather than getting lost in the visual details. This means creating a simple and clear layout that clearly communicates the main features and content of the website or application.

As you move forward with the design process, you can use the wireframe as a guide to build out the more detailed visual elements and user interface components. This approach can save you time and help in the case of a web development company, avoid making costly design mistakes down the line.

Overall, building a wireframe is an essential step in the development process of any application or website, and it can help you create a more effective and user-friendly product in the long run.

Wireframing is the process of creating a rough sketch or outline of a design for a website, app, or other digital product. It is typically done in the early stages of the design process to help visualize and organize the layout and structure of the product.

In the case of a company Wireframes can help designers and developers collaborate and communicate more effectively, as they provide a clear visual representation of the product's structure and functionality. They can also help stakeholders and clients better understand and provide feedback on the design before it is fully developed.

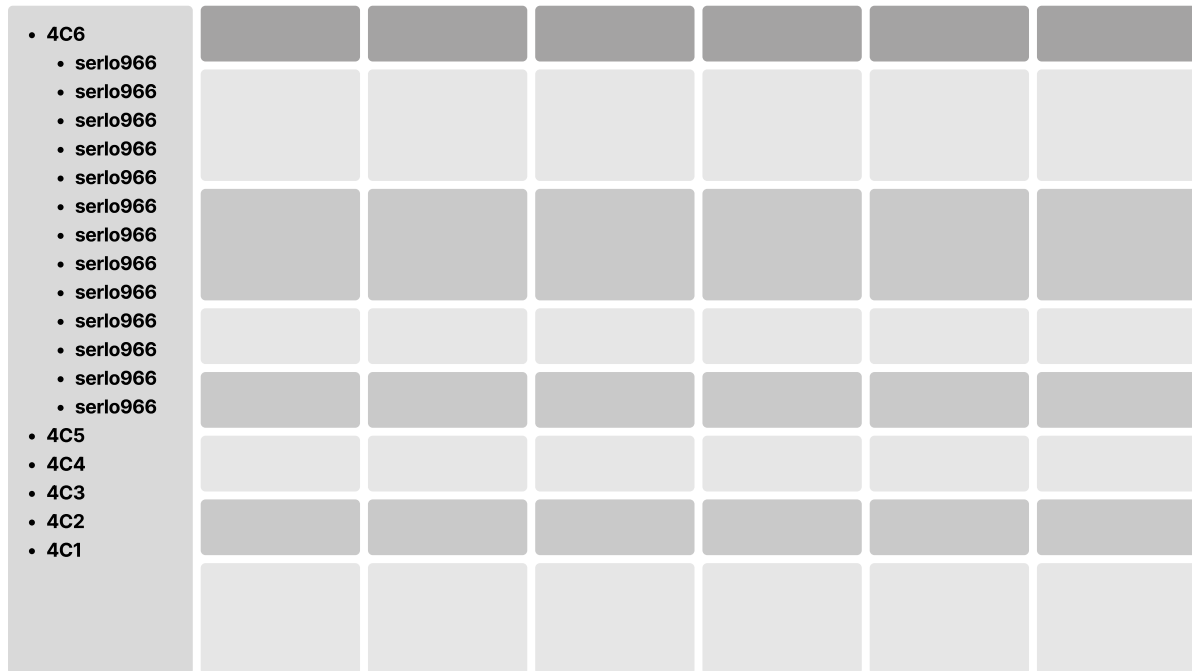
Wireframing is in my opinion one of the most important steps parts of the design process that can help streamline the development of a digital product by providing a clear and structured plan for the project.

To maybe understand the scale of a Figma file when produce, here is an example wireframe of a site, sidenote every page and frame you see in an app can be a Figma frame and has to be created:

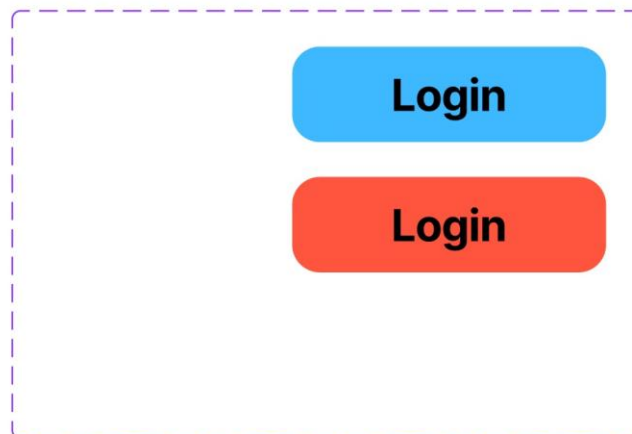
<https://www.figma.com/community/file/1115280671580274545>

First Versions of the Designs

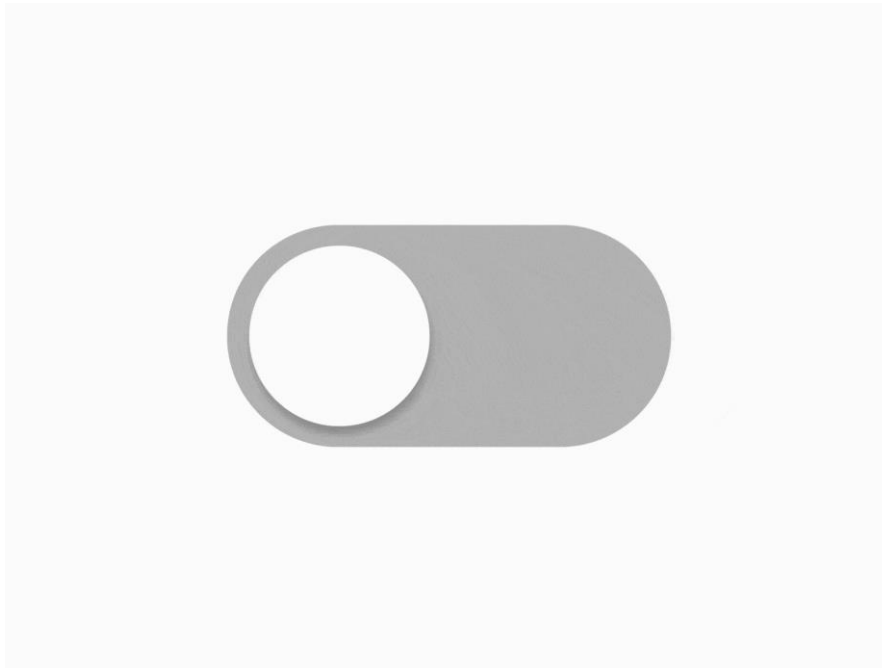
So, in the first versions I focused on the part that the user is most confronted with. With that I mean the timetable and menu for getting information
In the early stages this timetable/menu looked like this:



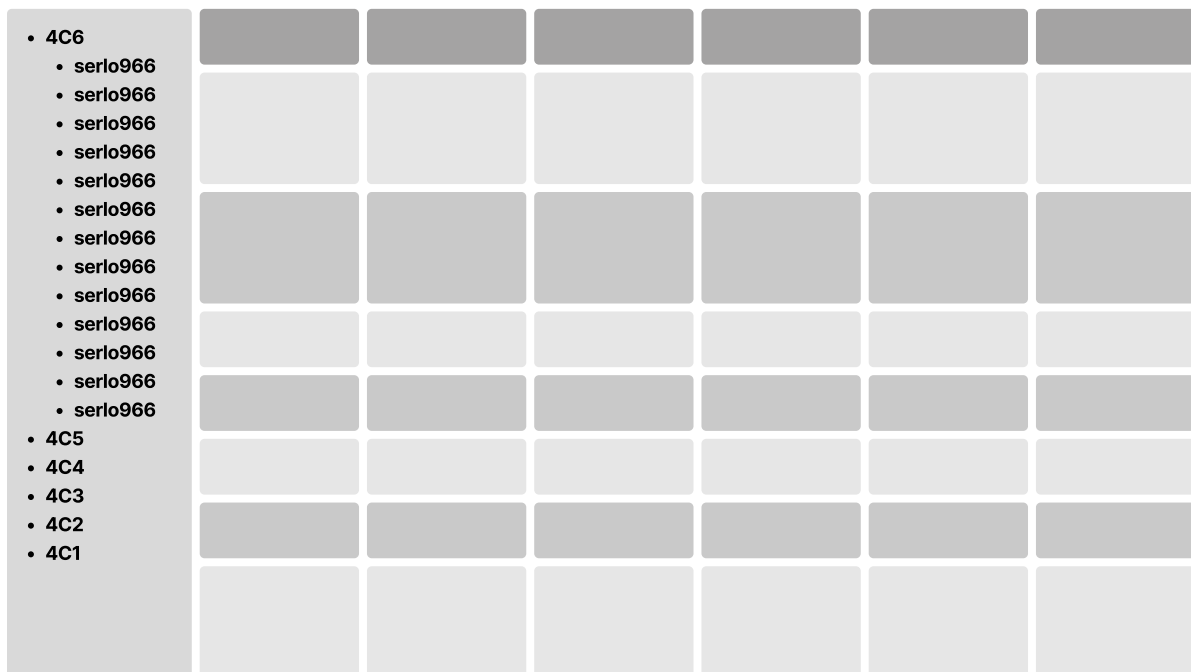
As I was building my project, I discovered a useful feature of Figma called 'components'. Components are reusable elements that can be used throughout the project. One example of a component is the navigation bar or 'navbar' that appears on top of a website to help users navigate. Rather than copying and pasting it from one page to another, a component can be created and used where and when needed. If changes need to be made, simply adjust the 'root' component and all other components will update automatically. Components can also have variations, such as a button in both red and blue colours.



With a combination of animations, links, and components you can create beautiful sites. For example, these buttons are created with Figma:



This was the first draft of how the calendar module itself should look like and by the way this was the first thing I created with Figma. Due to my experience in Illustrator creating this felt familiar to using the Adobe Vector editor. Here is how it looked like:



After this version I added a navigational bar which was at first not filled and a version of the design where the calendar is filled in with my credentials.

		Lundi	Mardi	Mercredi	Jeudi	Vendredi
<ul style="list-style-type: none"> • 4C6 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • 4C5 • 4C4 • 4C3 • 4C2 • 4C1 	8:10 - 9:40	mathe	english	sposa	mathe	scite
	10:10 - 11:40	lite	vieso	chelsea	français	daitsch
	11:45 - 12:30	etude	etude	chelsea		
	12:30 - 13:15	etude	etude	chelsea		
	13:15 - 14:00				lite	etude
	14:00 - 14:45				lite	etude
	14:50 - 16:20		français	scite	artso	english

The versions afterwards included the creation of components and having everything organized.

Sergonne Lou
iam: serlo966
entreprises: Chelsea, Lite
heures: 4, 4
salle: B.1.12, B.2.12
fraistonnnes: meindes 1. Kuer, dendes 2. Kuer
haus: Larochette
classe: 4C6

Page to view students information.

presents	salle	iam	other infos
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	

This is a version of the class overview.

Then the versions after this I designed the pages for showing a class and for showing the data for a specific student. These designs or wireframes were created with the help of head of Chelsea Studios, Alex who told me what they would need on a regular basis and confirmed me afterwards if the layout is comprehensible.

For more detail in my GitHub repository und the folder “figma_versions” you can find whole pdfs of the versions etc.

Here is a link to the GitHub folder:

https://github.com/uncreative/calendar/tree/main/figma_versions

Versions afterwards

The versions afterwards began to grow and grow and to declutter it I have organized the pages in 4 Categories, they are as following:

- Who is here right now?
- Who is when and who?
- Free Time

As already stated, the wireframe began to take form and here is how it started to look:

Calendar

iam: SerLo966
 Nom: Sergonne Lou
 Classe: 4C6
 Maison: Larochette
 Entreprise(s): Chelsea, Lite
 Salle: B.1.12, B.2.12

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
10:10 - 11:40	Fraistonn				
14:50 - 16:20			Fraistonn		

Interface to display students information.

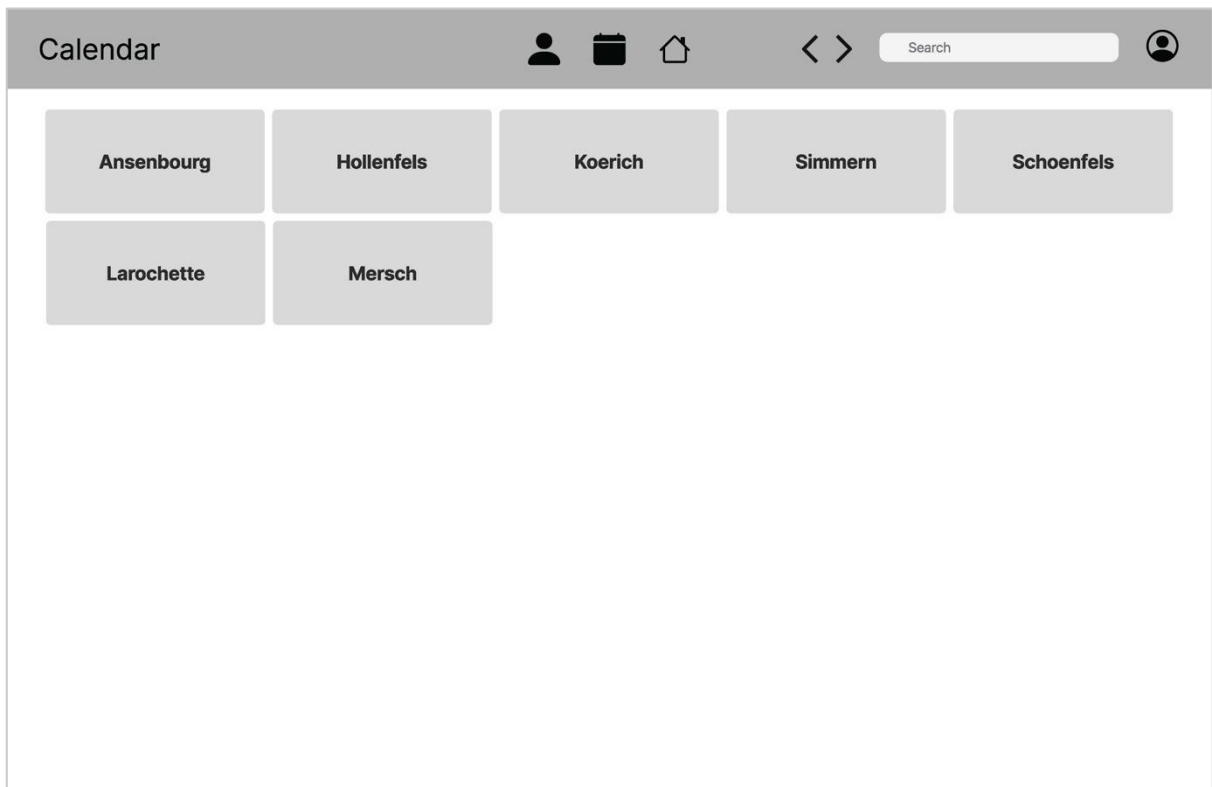
Calendar

presents	salle	iam	other infos
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	
Sergonne Lou	B.1.12	serlo966	

Overview of students in a class



Navigation menu for selecting classes.



Navigation menu for viewing houses to select either students or classes.

Calendar

Calendar

Search

		Lundi	Mardi	Mercredi	Jeudi	Vendredi
<ul style="list-style-type: none"> • 4C6 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • serlo966 • 4C5 • 4C4 • 4C3 • 4C2 • 4C1 	8:10 - 9:40	Math	English	Sposa	Math	SciTe
	10:10- 11:40	Lite	Vieso	Chelsea	French	German
	11:45 - 12:30	Etude		Chelsea		
	12:30 - 13:15			Chelsea		
	13:15 - 14:00		Etude		Lite	Etude
	14:00 - 14:45		Etude		Lite	Etude
	14:50 - 16:20		French	SciTe	ArtSo	English

General overview

Calendar

Calendar

Search

élèves	heures	iam	salle
sergonne lou	chelsea, lite	serlo966	B.1.12, B.2.12
sven bordez	chelsea, dragons	borsv	B.1.12,
sergonne lou	8	serlo966	B.1.12
sergonne lou	4	serlo966	B.1.12
sergonne lou	8	serlo966	B.1.12
sergonne lou	4	serlo966	B.1.12
sergonne lou	8	serlo966	B.1.12

View of classes filled in with my credentials.

The parts like the schedule are composed of these squares, which are components with variations. These look like this:

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
8:10 - 9:40					
10:10 - 11:40					
11:45 - 12:30					
12:30 - 13:15					
13:15 - 14:00					
14:00 - 14:45					
14:50 - 16:20					

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
10:10 - 11:40					
14:50 - 16:20					

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
10:10 - 11:40				—	—
11:45 - 12:30					
12:30 - 13:15					
13:15 - 14:00	—				
14:00 - 14:45	—				
14:50 - 16:20	—	—	—		

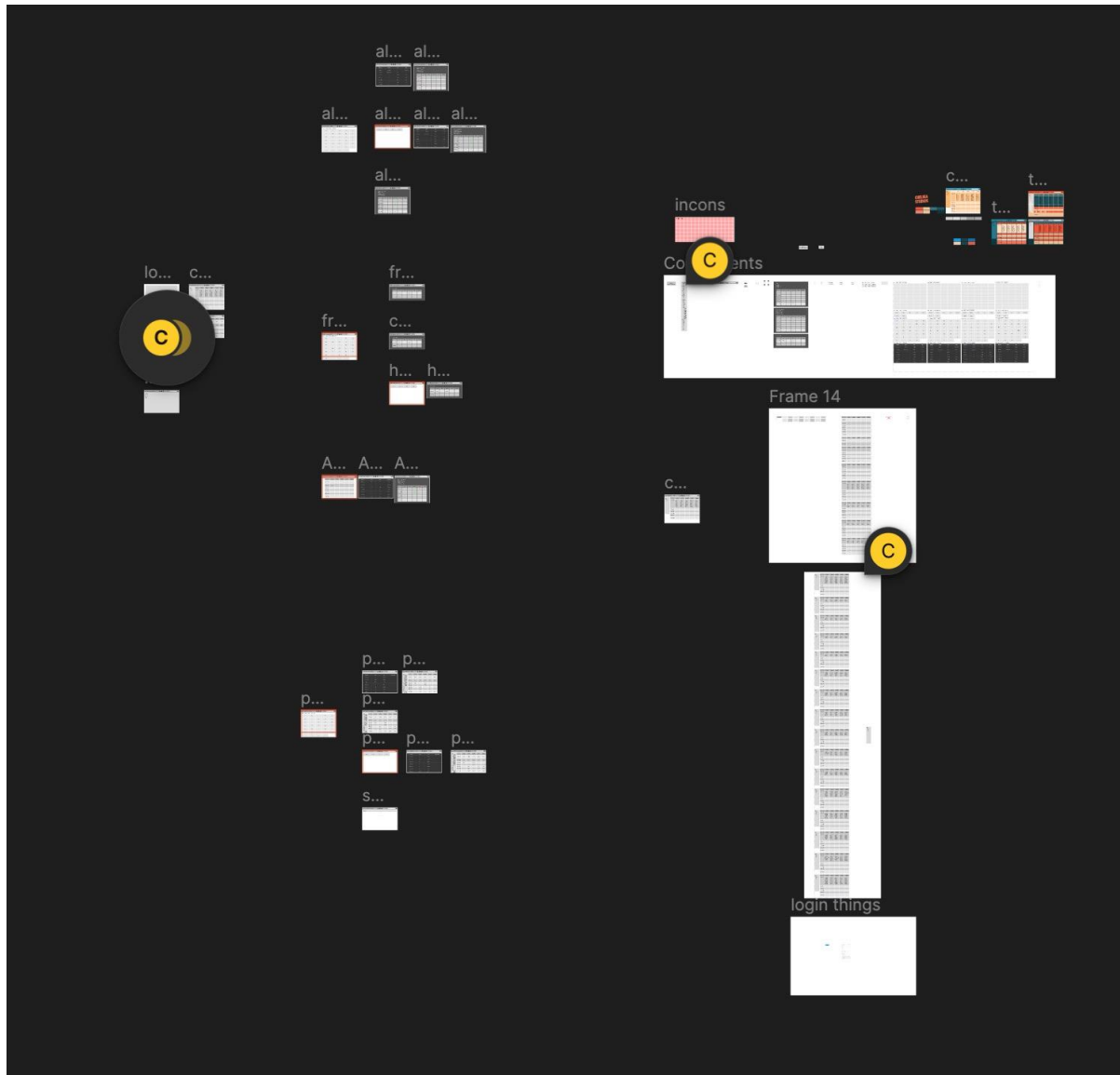
The components are combined in the dotted box and each occurrence in these boxes is a variation of the component.

Final Version

The final version is accessible under this link or as images down below:

<https://www.figma.com/file/TbWNUq76px77vzYljUtDYK/calendar>

<https://www.figma.com/proto/TbWNUq76px77vzYljUtDYK/calendar>



The part on the left are all the pages that have been shown off previously, visually divided in 4 sections.

- The section closest to the left border of the image are things like the settings page, the login and sign out page.
- The top “group” is the group for regrouping all students.
- The group below it is the group where you can view the free hours of the students.
- The lowest group are the frames responsible for detailed information on specific students.

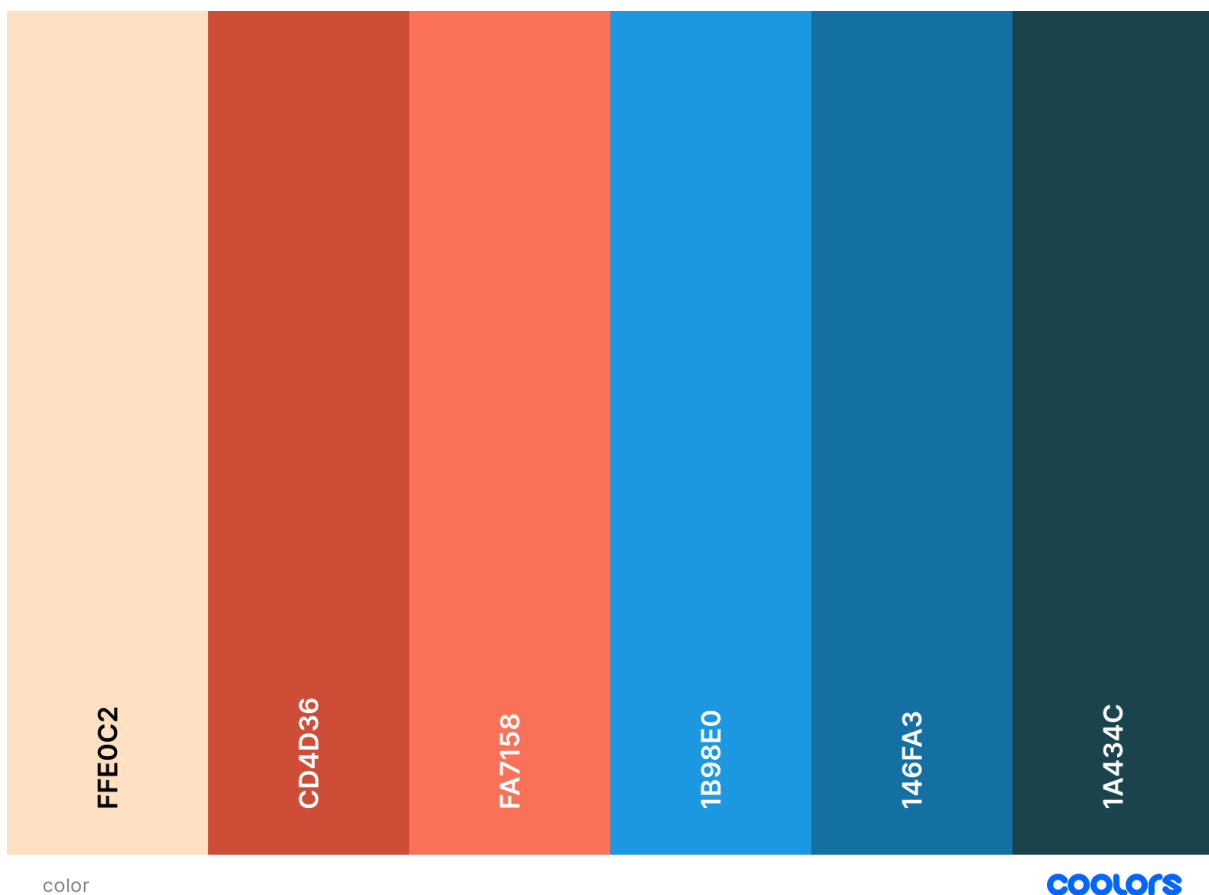
There is also a part of the very right, bigger than the part of the pages, this are the components I talked about earlier, and the little colourful part is the part where I tested everything coloured until I then made a fully coloured version.

Coloured version

After I somewhat finished the wireframe, I began to colour the frames, the plan is that every enterprise has its own colour scheme, and depending on who logs in the colours change appropriately. I started by doing the scheme for Chelsea, where the main colours related with cyan, and orange based on their logo.

The logo for Chelsea Studios is displayed in a bold, 3D-style font. The word "CHELSEA" is on the top line and "STUDIOS" is on the bottom line. The letters are primarily orange with a slight gradient and a dark shadow underneath, giving them a three-dimensional appearance.

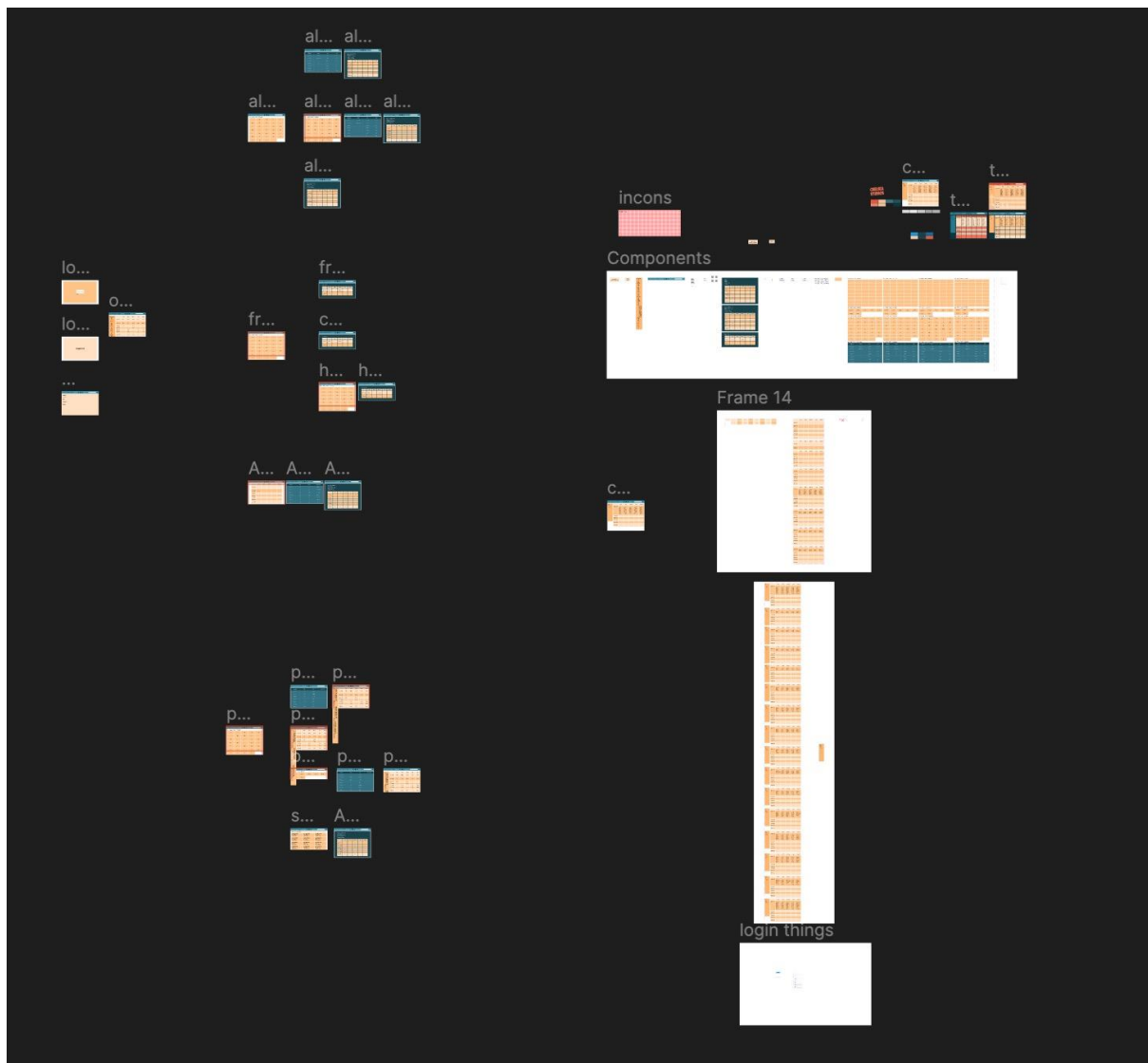
Here is their logo.



Here is the colorscheme I created.

I generated a colour scheme online with “Colors” and based on this I collared the frames by modifying the basic components that everything is composed of.

The current version looks like this:



And here is a link to explore that wireframe in more detail:

<https://www.figma.com/file/EPg6tPCdESPjdGQn545LW6/calendar-colors>

The colours might not be optimal, and I discarded that idea for now, because I am not a UI designer and didn't want to invest a lot of time for the colouring, that is a task for the future where I might ask another person for help.

Phase 2: Database Setup

The Database, the heart of my program was the second big thing I had to learn and setup. I decided to use a MySQL database visualized with PhpMyAdmin. I took this choice because the software I used to localhost my sites (localhost as the name suggests is a process hosted locally, this can be a database, a webserver, website etc.), had a MySQL database included. This software is called MAMP and included a MySQL database and also SQLite database.

Database client

As stated earlier I use PhpMyAdmin, a tool for managing MySQL Databases, as database client.

You can access a demonstration of a PhpMyAdmin system under:

<http://sergonnelou.ddns.net/phpmyadmin/>

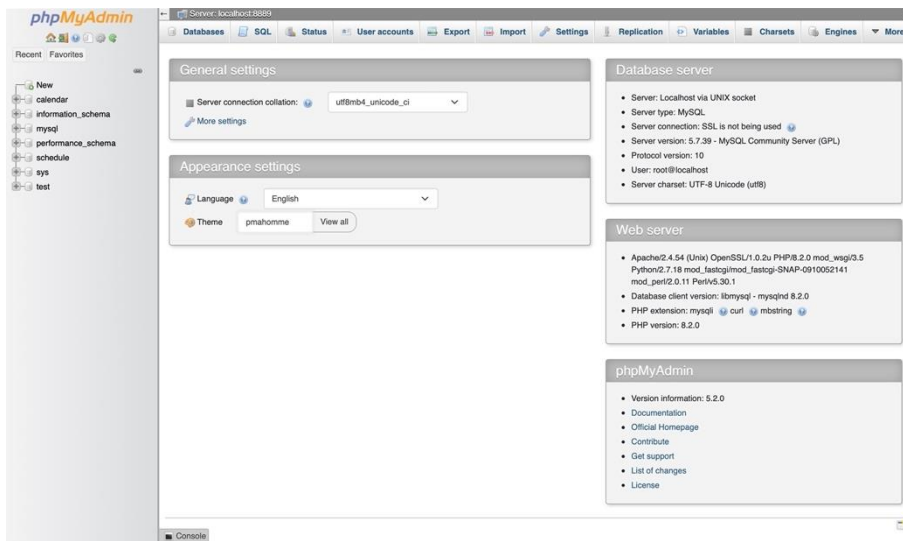
There you can login and explore the interface, there is a database called schedule and when you select it you can view one recent version of the database I actually used.



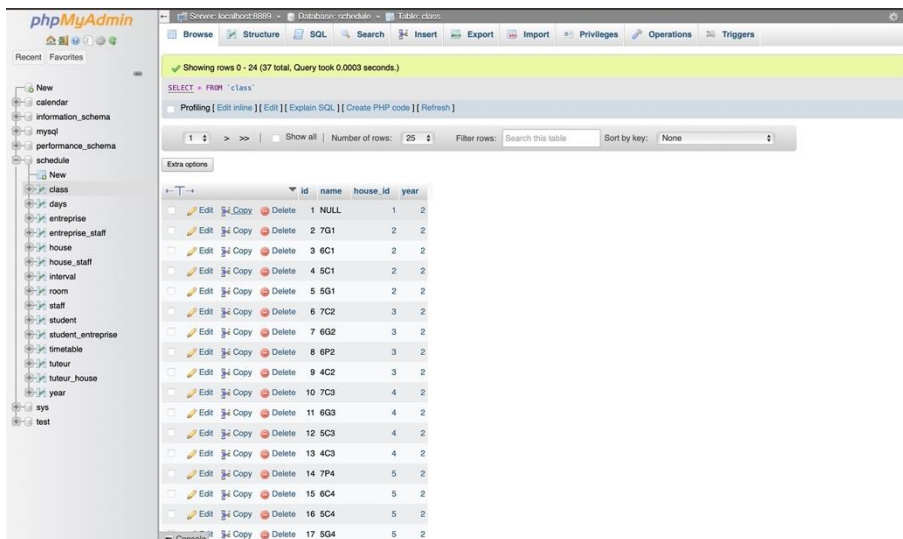
You will be greeted with this login page where you can enter the credentials.

As username as the picture above states is “visitor” and the password for the PhpMyAdmin client is also “visitor”.

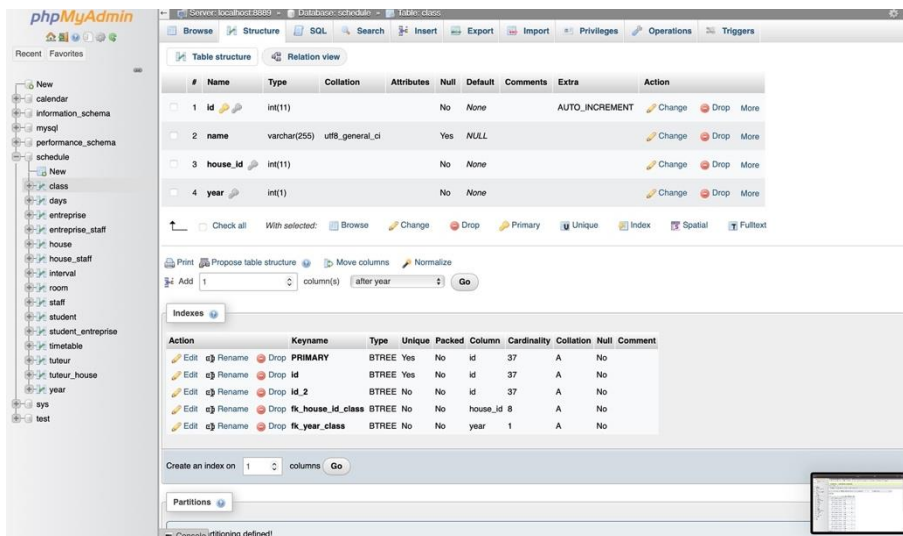
Here is how to GUI of PhpMyAdmin, looks like resumed in some pictures:



This is the home screen.



And this is the GUI for viewing the contents of a table.



Here is the structure of the table visualised.

This database client brings a lot of advantages with it, such as the visual editor of the structure, the possibility to generate SQL from the GUI (editing a row within the GUI, etc.), creating, dropping, and truncating tables, ...

The initial version of PhpMyAdmin was written in PHP and was designed to provide an easy-to-use interface for managing MySQL databases over the web. It quickly gained popularity among web developers, particularly those who were not familiar with command-line tools like the MySQL command line.

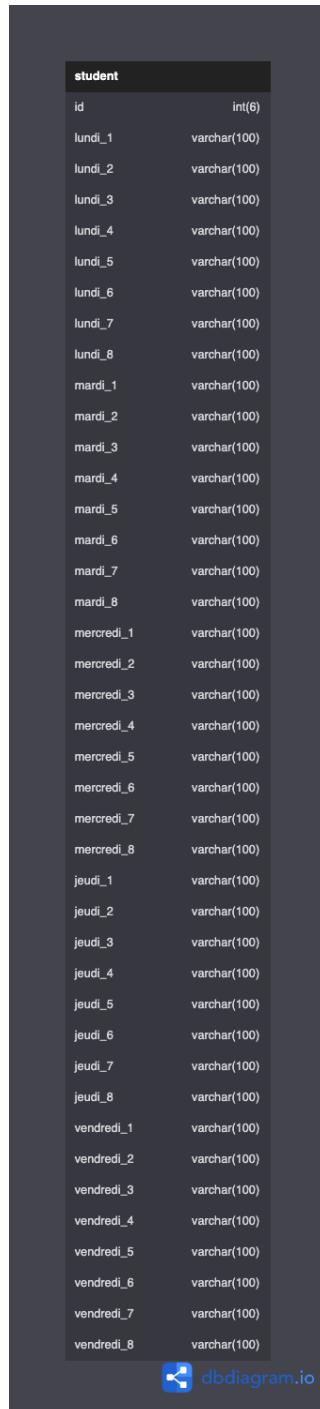
Over the years, PhpMyAdmin has undergone significant development and improvement, with new features added and bugs fixed in each new version. Some of the notable features of PhpMyAdmin are:

1. A User-friendly interface: PhpMyAdmin has a simple and intuitive interface that allows users to manage their MySQL databases and tables with ease.
2. Database management: With PhpMyAdmin, users can create, modify, and delete databases, tables, fields, and indexes, as well as import and export data.
3. SQL query execution: PhpMyAdmin allows users to execute SQL queries directly in the web interface, making it easy to perform complex database operations.
4. Database User management: PhpMyAdmin provides a user management interface that allows administrators to add, modify, and delete database users and set their privileges.
5. (Server monitoring: PhpMyAdmin can be used to monitor MySQL server status, as well as view server variables and processes.)

Today, PhpMyAdmin is used by a lot of web developers and administrators around the world and even the Enterprise Lite is using it and was using it before I discovered that they used it. To no surprise PhpMyAdmin is widely regarded as one of the best MySQL management tools available for free and opensource.

First encounter with a Database

The structure of my database was at first eyeballed, I used varchar, a datatype in a database, to store html code that would be inserted afterwards, the structure was very primitive and wasn't going to work for the project.



The image shows a screenshot of a database table structure for a table named 'student'. The table has the following fields:

student	
id	int(6)
lundi_1	varchar(100)
lundi_2	varchar(100)
lundi_3	varchar(100)
lundi_4	varchar(100)
lundi_5	varchar(100)
lundi_6	varchar(100)
lundi_7	varchar(100)
lundi_8	varchar(100)
mardi_1	varchar(100)
mardi_2	varchar(100)
mardi_3	varchar(100)
mardi_4	varchar(100)
mardi_5	varchar(100)
mardi_6	varchar(100)
mardi_7	varchar(100)
mardi_8	varchar(100)
mercredi_1	varchar(100)
mercredi_2	varchar(100)
mercredi_3	varchar(100)
mercredi_4	varchar(100)
mercredi_5	varchar(100)
mercredi_6	varchar(100)
mercredi_7	varchar(100)
mercredi_8	varchar(100)
jeudi_1	varchar(100)
jeudi_2	varchar(100)
jeudi_3	varchar(100)
jeudi_4	varchar(100)
jeudi_5	varchar(100)
jeudi_6	varchar(100)
jeudi_7	varchar(100)
jeudi_8	varchar(100)
vendredi_1	varchar(100)
vendredi_2	varchar(100)
vendredi_3	varchar(100)
vendredi_4	varchar(100)
vendredi_5	varchar(100)
vendredi_6	varchar(100)
vendredi_7	varchar(100)
vendredi_8	varchar(100)

The screenshot also includes a logo for dbdiagram.io at the bottom right.

As you can see, the structure here has a lot of room to improve. As well as not having all the details needed this structure had no real future due to the massive amounts of fields.

In my first prototype I started to learn about Databases. I got used to SQL (structured query language) and began to understand the very basics, how to connect to a database and all the other basics.

To understand what's following in the next chapters, one has to understand the basics which aren't that complicated on the surface. Explaining how the magic works in the background another whole TraPe could be.

The basics

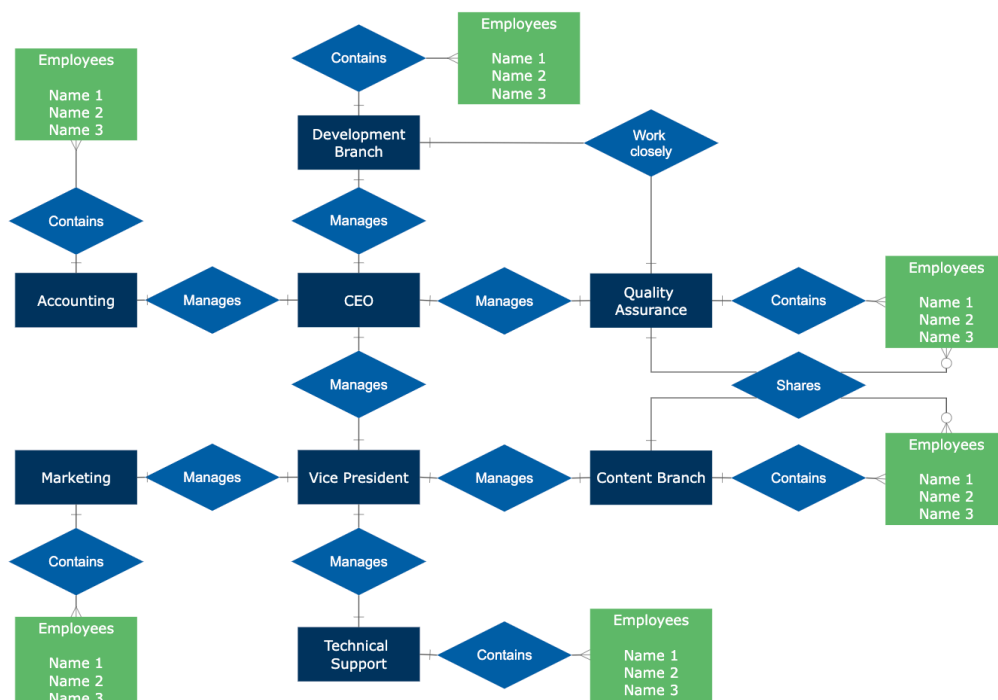
A database is a structured collection of data that is stored and organized in a specific way, making it easy to search, access, and manage.

An example of a database is a library catalogue. A library catalogue organizes information about books, and other materials that are available for borrowing. The catalogue typically contains information about the author, title, publisher, publication date, and subject matter of each item in the library's collection.

The catalogue can be searched by various criteria, such as author, title, keyword, or subject, making it easy for library stuff to find the materials they need. This is an analogue Database; another way of picturing databases is a combination of a lot of excel sheets.

There are databases all around you, for example a parking house uses a database to store when a car has entered and when a car wants to exit accesses the information. Every messenger app is a database overlay basically and simply put.

Entity Relationship Diagram - Department Relationships



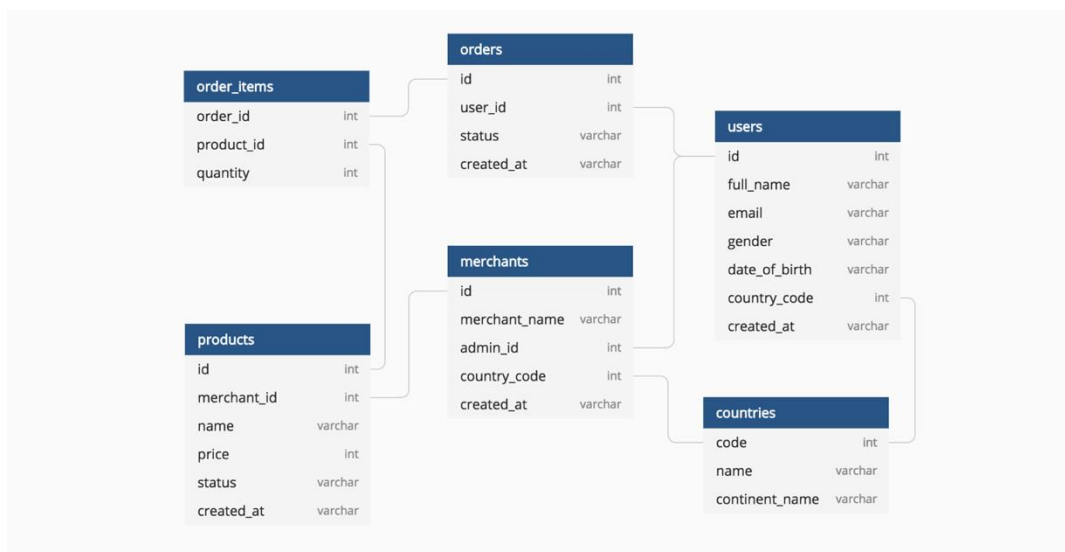
Database layout example

Basic Structure of a Database

The structure of a database can be broken down into three main components: tables, fields, and records.

A table is a collection of related data that is organized into rows (also called records) and columns (also called fields). Each table has a name and is made up of one or more fields. In a database, there can be multiple tables, each containing data about a specific subject or entity.

A field is a single piece of data within a table. Each column in a table represents a field, and each field has a unique name and data type. For example, a field in a student table could be "first name," "last name," or "IAM."



In this example, `order_items` are the table name, `order_id`, `product_id` and `quantity` are fields or also known as columns.

Overall, the structure of a database is designed to organize and store data in a logical and efficient way, making it easy to search, access, and manipulate the information.

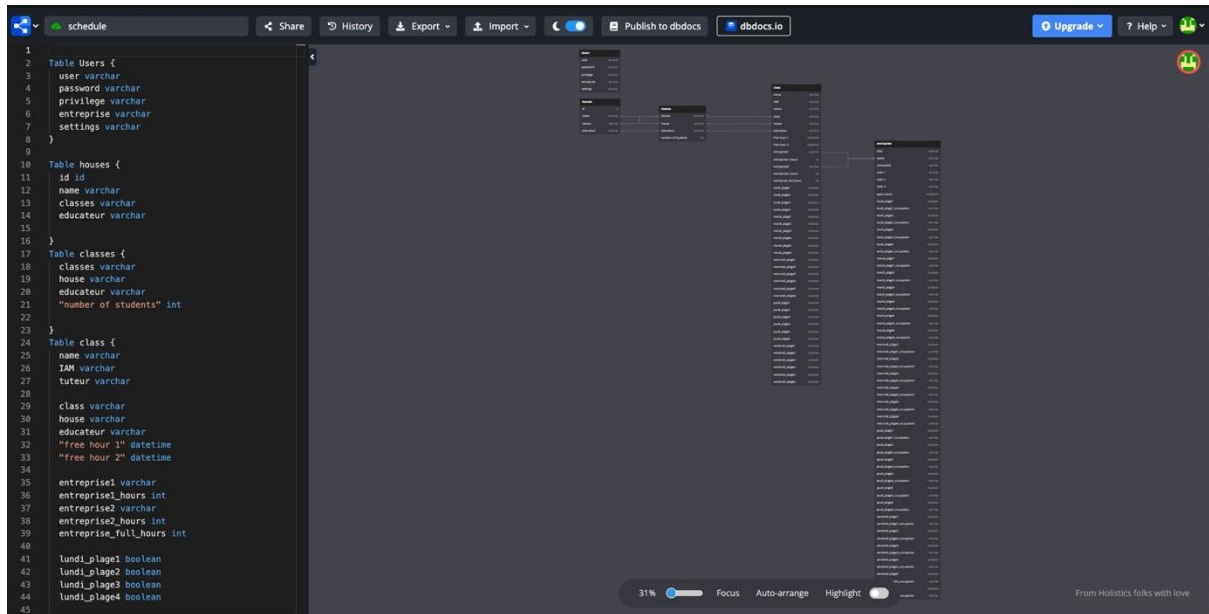
id	name
1	Lundi
2	Mardi
3	Mercredi
4	Jeudi
5	Vendredi

This is another way to visualize a table in a database.

With that knowledge we can continue our journey to build up a database structure. To understand more you can navigate to the ending where there is a definition chapter about databases.

Second try of a database

After learning all these basics by a lot of trial and error and googling, I started to plan my second try to create a database. To make the whole thing less unorganised I used the same principle that I used to plan my site, in this case it wasn't Figma but dbdiagramm.io to create a database structure. Although there are also other tools like Lucidchart, DrawSQL, Diagrams.net, etc.

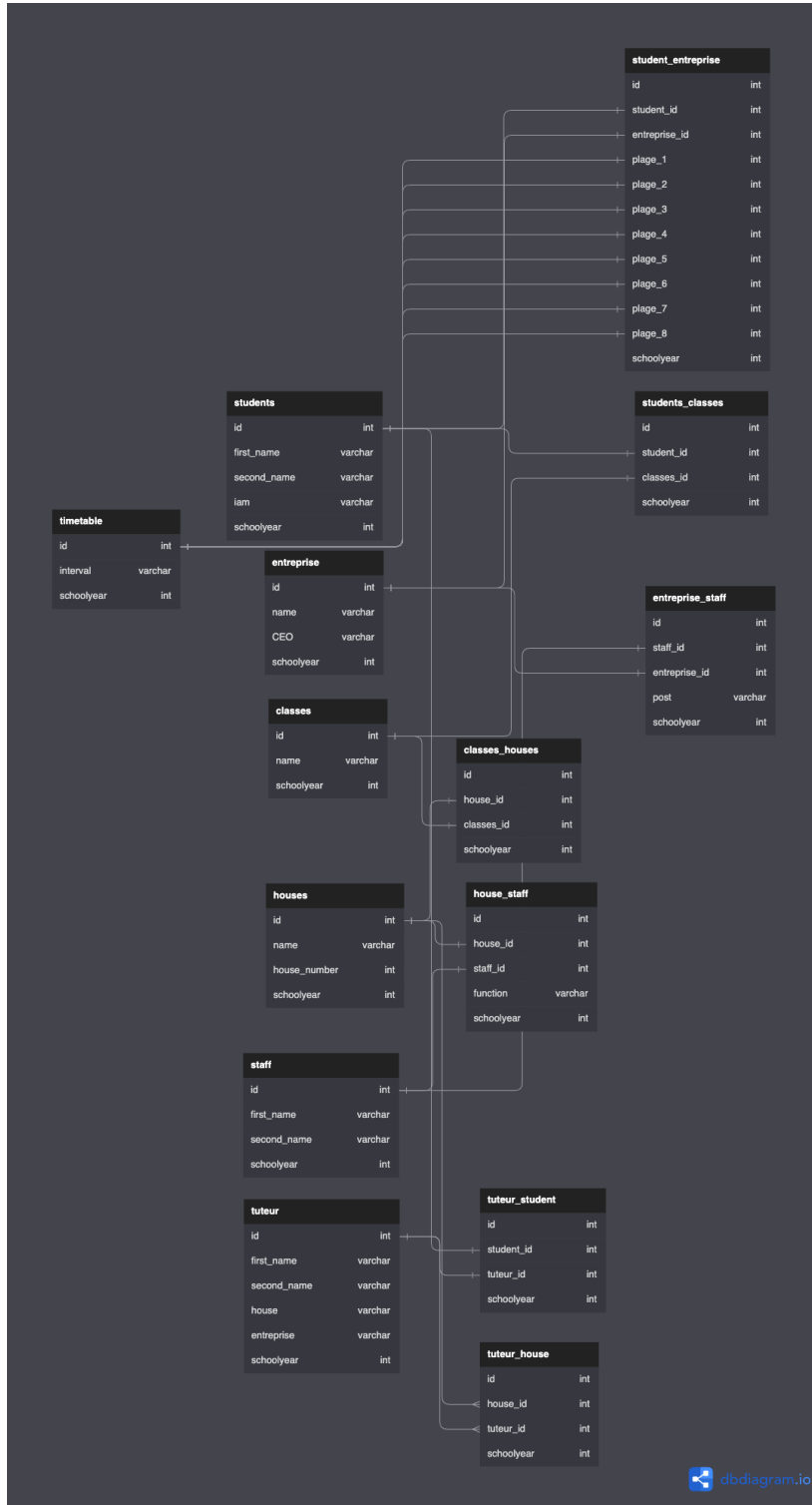


Here you can view the GUI of the site.

Here you can visit the structure:

<https://dbdiagram.io/d/636cf89ac9abfc611171b41b>

And here is a picture of it:



To organise relationships such as in which enterprise a student is I used connecting databases, one example of this is the table `student_entreprise` built up like this:

Database: `schedule`, Table: `student_entreprise`, Purpose: Dumping data

id	student_id	entreprise_id	room	plage_1	plage_2	plage_3	plage_4	plage_5	plage_6	plage_7	plage_8	year
----	------------	---------------	------	---------	---------	---------	---------	---------	---------	---------	---------	------

Where I used the Ids of the classes and Students to link them together, the student with the "Id 1" is in enterprise with the "Id 2" and has as "plage 1" in the enterprise this plage, also represented as ID.

I took the structure from the site and created it in MySQL. In other words, I exported it as SQL and then imported or executed it in my database.

Here is how that MySQL file looked like:

<https://pastebin.com/uy6gxds3>

And here is a brief preview of the contents of the SQL:

```
--
-- Table structure for table `classes`
--
CREATE TABLE `classes` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `schoolyear` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Table structure for table `classes_houses`
--
CREATE TABLE `classes_houses` (
  `id` int(11) DEFAULT NULL,
  `house_id` int(11) DEFAULT NULL,
  `classes_id` int(11) DEFAULT NULL,
  `schoolyear` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Table structure for table `entreprise`
--
CREATE TABLE `entreprise` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  `CEO` varchar(255) DEFAULT NULL,
  `schoolyear` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----
```

Optimising the database

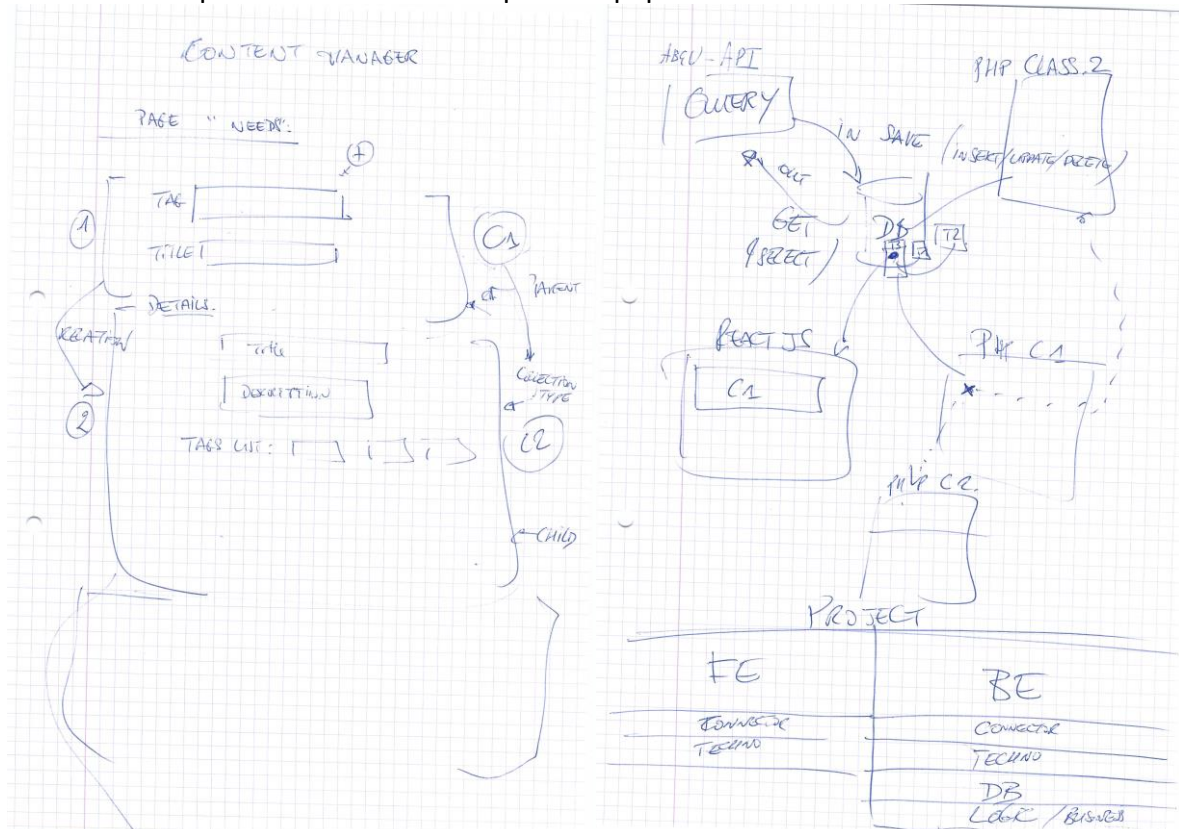
This database before was based on some old standards or inefficient methods. I made a Stage at Nvision where I worked with the programmers to test out new systems, and there I asked what I could ameliorate, and they could help me a lot.

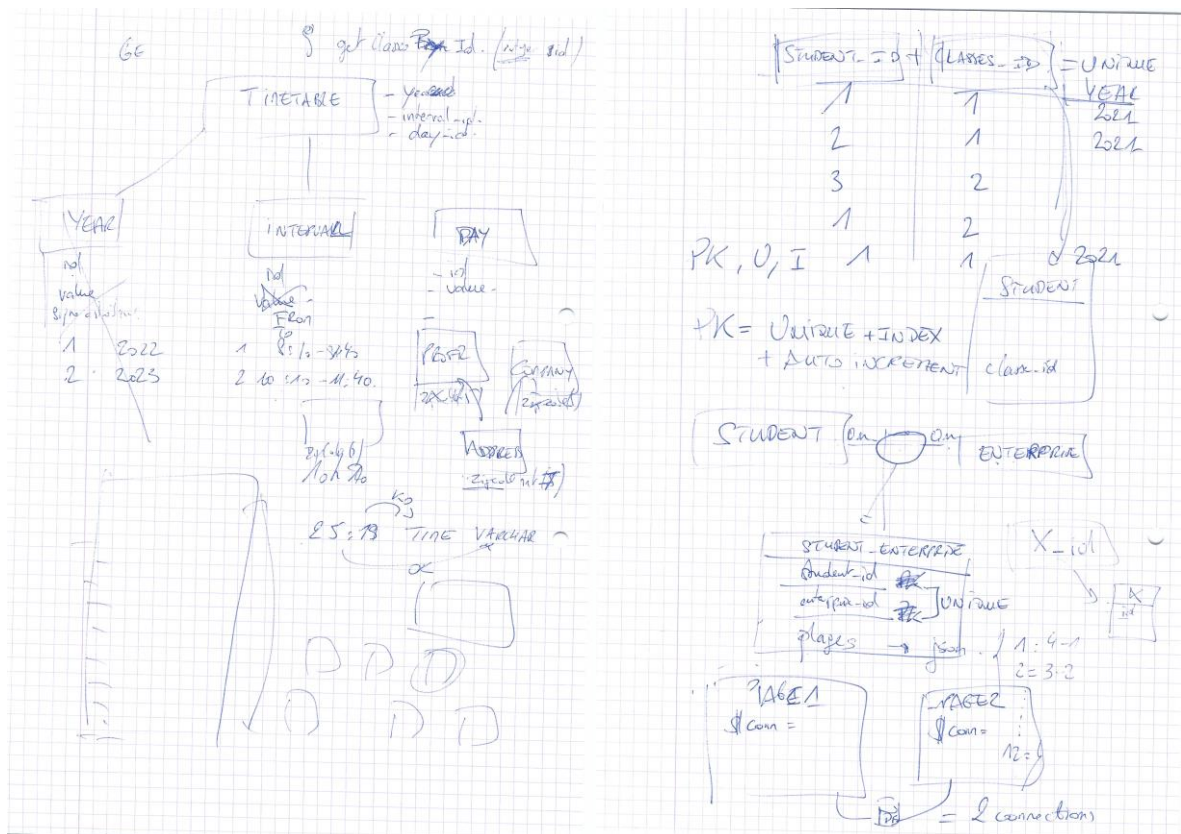
Here are my notes from the conversation with their lead developer an CIO:

This is the part that I wrote down on my notes app.

```
log db
truncate all existing data.
--
db: class
    rename, classes to class
    rename schoolyear to year
    set primary to id and set. id unique
rename. bunch of stuff for singular
--
```

And this is the part that I scribbled on piece of paper.





I took a lot of things from the conversation with the head developer. The main points were, that everywhere where you have doubled data is a bad designed database, there should in no case be doubled data because it is inefficient and decreases the speed of a database. The other main thing I took away from it, was that foreign keys etc are important to implement and they are there in order to keep data integrity there.

As I implemented the ameliorations, that include Primary keys, indexes, foreign keys etc. I had to replace all the existing PHP code because a lot has changed, but I didn't want to do that by hand, so I Decided to make a little python script that generates me an PHP structure and everything I need but more on this program later.

Mini automation project for PHP structure

So as explained earlier I wrote a simple script to automate the creation of the PHP part of the code. This way due to laziness and the lack of motivation to change every insert statement and adjust everything with copy and pasting code. So, I thought why not take the structure of the database and export it into files and convert them into insert statements and get statements which will make my life a lot easier.

The script lets a popup appear where you can select your contents that you want and the if you press generate PHP you get the pre generated PHP to your clipboard.



Here is that popup.

The script behind it is simple and it contains three parts.

The first part, the initialization looks like this:

```

from tkinterdnd2 import *
from tkinter import filedialog, ttk
from tkinter.ttk import *
from datetime import datetime
from tkinter import *

root = TkinterDnD.Tk()
root.title('menu')
root.geometry("1000x300")

global path_entreprise, value_entreprise, intvar_entreprise

path_entreprise =
"/Users/lousergonne/Documents/GitHub/calendar/database/new_struct/sql_structure/php_insert/entre
prise"
value_entreprise = ["FALSE"]

def change_value_entreprise():
    global path_entreprise, value_entreprise, intvar_entreprise

    print("entreprise")
    if value_entreprise[-1] == "FALSE":
        print("entreprise -> TRUE")
        print(value_entreprise)
        value_entreprise.append("TRUE")
        print(value_entreprise)

    elif value_entreprise[-1] == "TRUE":
        print("entreprise -> FALSE")
        print(value_entreprise)
        value_entreprise.append("FALSE")
        print(value_entreprise)

intvar_entreprise = IntVar()
Checkbutton(root, text="entreprise", variable=intvar_entreprise,
command=change_value_entreprise, pady=-1, padx=-1).grid(column=1, row=0)

```

And this is repeated on and on and on. Here are all the functions and pieces of code in the document.

<https://pastebin.com/u59iXpZt>

The last part of the script executes everything and combines it:

```
import subprocess

def write_to_clipboard(output):
    process = subprocess.Popen(
        'pbcopy', env={'LANG': 'en_US.UTF-8'}, stdin=subprocess.PIPE)
    process.communicate(output.encode('utf-8'))

def generate():
    result = []
    struct_begin = """<?php

$db_host = 'localhost';
$db_user = 'root';
$db_password = 'root';
$db_db = 'schedule';

$mysqli = @new mysqli(
    $db_host,
    $db_user,
    $db_password,
    $db_db
);

if ($mysqli->connect_error) {
    echo 'Errno: ' . $mysqli->connect_errno;
    echo '<br>';
    echo 'Error: ' . $mysqli->connect_error;
    exit();
}"""
    struct_end = """$mysqli->close();
?>
"""
    result_cache = []

    for count, value in enumerate(values):
        # print(value)
        if value[-1] == "TRUE":
            print("values", values_path[count])
            with open(values_path[count]) as f:
                result_cache.append(f.read())
        else:
            pass

    print(result_cache)
    result = struct_begin + " ".join(result_cache) + struct_end
    print(result)
    write_to_clipboard(result)

ttk.Button(root, text="Generate php filler", command=generate).grid(column=1, row=16)

root.mainloop()
```

And here is the whole file combined:

<https://pastebin.com/K6ZStj4y>

This massive amount of data hasn't created itself, and the script part neither. I used a lot of helper scripts to format everything appropriately.

Here for example is the script that created all the import statements in SQL:

```
import sys, os, re
os.mkdir("/Users/lousergonne/Documents/GitHub/calendar/database/new_struct/sql_structure/php_insert/")
with
open("/Users/lousergonne/Documents/GitHub/calendar/database/new_struct/sql_structure/query",
"r") as f:
    fr = f.read()
    fr = fr.split("\n")
    for insert_statement in fr:
        insert_statement = insert_statement.replace("INSERT INTO `", "")

        name = insert_statement.split('`', 1)[0]
        print(name)

        parenthesis = re.search('(`(.*)`)', insert_statement)
        value_names = parenthesis[0]
        value_index = parenthesis[1]

        print("value_names",value_names)
        print("value_index",value_index)

        values = value_names.replace("`", "")
        values = values.replace("(", "")
        values = values.replace(" id", "id")

        # values = values.split(", ")
        print("values",values)
        values1 = values.split(", ")
        defining_values_lst = []
        ending_statement_lst = []
        field_name_array_lst = []

        num = 0
        for value in values1:
            if value == " id":
                value = "id"
            defining_values_lst.append(f"${name}_{value} = array();")
            ending_statement_lst.append(f"${name}_{value} = $result_{name}[{num}][1];")

            field_name_array_lst.append(f"" array("{value}", ${name}_{value}),""")

            num += 1

        defining_values = "\n".join(defining_values_lst)
        field_name_array = "\n".join(field_name_array_lst)
        ending_statement = "\n".join(ending_statement_lst)

        query = f"""
//import {name}

{defining_values}

$arr_{name} = array(
{field_name_array}
);

$sql_{name} = "SELECT {values} FROM {name}";

$result_{name} = import_arr($sql_{name}, $arr_{name});

{ending_statement}
"""
```

```
print(query)
with
open(f"/Users/lousergonne/Documents/GitHub/calendar/database/new_struct/sql_structure/php_insert
/{name}", "w") as f:
    f.write(query)

# name = ""
# defining_values = f" = array();"
# field_name_array = ""
# index = ""
# ending_statement = f"${name} = $result_{name}[{index}][1]"
```

All scripts are listed here:

A script to remove duplicates in a folder:

<https://pastebin.com/gFyu8EzW>

A script to remove everything in a folder except some files:

<https://pastebin.com/chdXeP38>

A script to replace something in filenames from a whole folder:

<https://pastebin.com/NgS7NnRx>

A script to create a file tree, was used for testing how to display the checkboxes:

<https://pastebin.com/Y6ca4c4M>

A script to import database things:

<https://pastebin.com/SZJKP784>

A script to organise files containing SQL queries and format them for further use:

<https://pastebin.com/vqMz7Vc4>

The same thing as above only for txt files:

<https://pastebin.com/Wzr1n7EF>

Phase 3: Coding

For the coding part I decided to use PHP for getting the data from the Database, HTML, JavaScript for the scripting and calculation processes and CSS for all the styling.

GitHub

My Project is all in one folder synced up with Git and GitHub, Git is a version control system for tracking changes in files and collaborating on projects and GitHub is a web-based hosting service for Git repositories that provides tools for collaboration, such as issue tracking and pull requests. Together, they allow developers to work on projects more efficiently and collaborate more effectively.

Here you can go view the GitHub repository:

<https://github.com/unkreative/calendar>

If you want, you can also look at other projects I've done so far under this link :

<https://github.com/unkreative>

Folder Structure

I used this system with GitHub because I work from multiple computers and have to have everything synced up and be the same at every computer, you could compare it to a Word document on the cloud. Another reason is that having a changelog is quite nice for debugging and logging purposes. This changelog makes it possible to show old versions of my code.

```

1. calendar
2. |— css
3. |— database
4. |   |— database_data
5. |   |— new_struct
6. |   |   |— input_data
7. |   |   |— insert query copies
8. |   |   |— sql_new_structure
9. |   |   |— sql_structure
10. |   |— queries
11. |   |   |— sql test data
12. |   |   |— sql_import
13. |— import_templates
14. |   |— import_all
15. |— old calendar stuff
16. |— pages
17. |— scripts
18. |   |— python
19. |— test purpose file

```

The GitHub repository includes 21 directories and 212 files that are important, the rest of it is NPM and testing stuff.

Here is a link to the file tree that also shows the files in the folders you can view at the top:

<https://pastebin.com/eeFuPbFq>

And here is a full file tree with everything from NPM packages to my code and so on:
<https://pastebin.com/2VUSxdA9>

The Project is structured as shown in the file tree above for example the folder Pages include the PHP files for Pages, here one example of such a file:
<https://pastebin.com/HzbZaryn>

Dimensions of the repository

To demonstrate the size of the repository where all my code is stored in, I added some statistics.

The word count of the folder where all my pages are is: 40.164 Words in total. For comparison my TraPe has around ~10.000 words and this is only one folder of many.

The whole GitHub directory including everything, from code to scripts to NPM (node package manager) modules consists of 12.601.514 words.

The whole GitHub without these NPM modules, has around 3.976.192 words. Including all the generated stuff for my little PHP generator etc.

To count these words, I did not dedicate an astounding amount of time in counting each word or even file. I simply asked ChatGPT to generate me a script and I then modified it to my liking.

```
import os

def count_words_in_file(file_path):
    try:
        with open(file_path, 'r') as file:
            return len(file.read().split())
    except:
        pass

to_add = []
def count_words_in_directory(directory_path):
    for root, dirs, files in os.walk(directory_path):
        for filename in files:
            if "test purpose file" in filename or "test purpose file" in root:
                pass
            elif "node_modules" in filename or "node_modules" in root:
                pass
            elif ".git" in filename or ".git" in root:
                pass
            else:
                file_path = os.path.join(root, filename)
                print(f"{file_path}: {count_words_in_file(file_path)} words")
                to_add.append(count_words_in_file(file_path))

# Replace the path with the directory you want to count the words in
directory_path = "/Users/lousergonne/Documents/GitHub/calendar"

count_words_in_directory(directory_path)

cache = 0
```

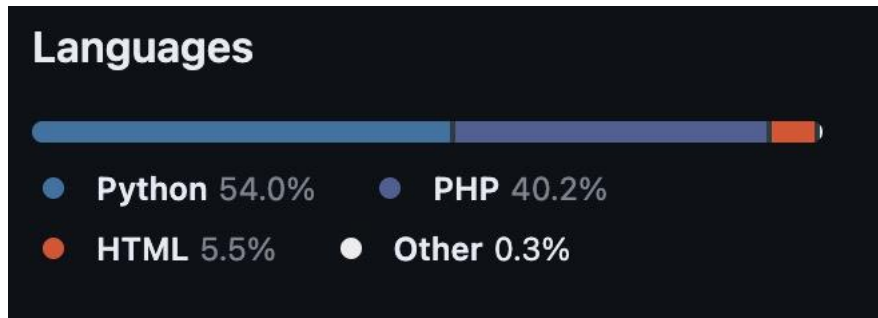


```

for x in to_add:
    try:
        cache = cache + x
    except:
        pass
print()
print(cache)

```

But for language usage in my projects, I used the GitHub statistics for this purpose. Here is the brief overview from GitHub:



A lot of the Python files are the scripts mentioned earlier.

Here is the full overview of files in the GitHub repository.

A table titled "Languages" showing the count of files for various languages in a GitHub repository. The table is as follows:

Language	Count
Markdown	2,196
TypeScript	345
XML	73
YAML	391
HTML	87
JSON	2,162
Less	105
Text	912
Ignore List	184
JavaScript	18,386

Here is the more detailed version of this, note that the 19.000 JavaScript files are from the node components.

Overview of Folder Contents

Scripts

What are those scripts?

The folder Scripts contains as the name suggests scripts for all kind of things, mostly JavaScript but also python. Here is one example of a script to insert the Navbar (Navigational Bar at the top of the site), because rather than copy and pasting code and when you change something you have to change it in every file, I tested an approach like components in Figma.

Contents of the folder

An example of the content of the folder Scripts is the script that I wrote that executes in the file if called and takes the code from an external file to make the navbar.

Contents of *navbar.js*

```
1. console.log("aa")
2. fetch("/pages/navbar.html")
3.   .then(response => response.text())
4.   // .then(text => console.log(text))
5.   .then(text => nav.insertAdjacentHTML("afterbegin", text))
6.
7. var nav = document.getElementById('navbar');
```

Important part of *navbar.html* (the code that is inserted in the pages)

<https://pastebin.com/N2bAda8X>

CSS

What is CSS?

CSS stands for Cascading Style Sheets, which is a language used to describe the appearance of web pages. This includes things like colours, fonts, spacing, animations, and layout.

To modify the appearance of HTML elements, CSS uses selectors to target those elements and apply styles to them. For example, you can create a custom tag like "container" and apply properties to all HTML objects with that tag. To include CSS in an HTML file you can import or create it with the style tag.

While CSS is a powerful tool for creating beautiful, responsive, and interactive web pages, it can be complex and challenging to use at times.

Folder contents

The Folder CSS includes all CSS file to centralize everything concerning CSS, Here can you see an example of such a file:

<https://pastebin.com/01V6VKJU>

This is a handwritten CSS file; therefore, it is kind of organized. If I would use a CSS framework a CSS file would look like this:

<https://pastebin.com/SEV1X3uF>

Database

In this folder I stored everything concerning databases, in there are the randomly generated data sets, the actual data from the real world, the structure of each table, insert queries for each table, etc.

Here is an example of such a SQL file:

```
insert into classes_houses (id, classes_id, house_id, schoolyear) values (1, 45, 20, 2021);
insert into classes_houses (id, classes_id, house_id, schoolyear) values (2, 43, 17, 2022);
insert into classes_houses (id, classes_id, house_id, schoolyear) values (3, 45, 7, 2021);
```

And here is the whole file on a Pastebin:

<https://pastebin.com/OpuRWYdM>

Insight and explanation of a piece of code

To give a bit of insight in the coding part of my Project, I will explain one whole Page, one page is represented by one file opened and processed in the browser (this is in my case, there are also other methods), so here is the code behind the page that shows the classes: The code is divided in two parts, the PHP side, and the HTML side.

This is the PHP part:

```
<?php

$db_host = 'localhost';
$db_user = 'root';
$db_password = 'root';
$db_db = 'schedule';

$mysqli = @new mysqli(
    $db_host,
    $db_user,
    $db_password,
    $db_db
);

if ($mysqli->connect_error) {
    echo 'Errno: ' . $mysqli->connect_errno;
    echo '<br>';
    echo 'Error: ' . $mysqli->connect_error;
    exit();
}

//echo 'Success: A proper connection to MySQL was made.';
//echo '<br>';
//echo 'Host information: ' . $mysqli->host_info;
//echo '<br>';
//echo 'Protocol version: ' . $mysqli->protocol_version;

$link_value = $_GET['classes_id'];
//echo $link_value;

$id = array();
$class_id = array();
$student_id = array();
```

```

$sql_selector = "SELECT id, student_id, classes_id FROM students_classes WHERE classes_id = " .
$link_value . ";";

$result_selector = $mysqli->query($sql_selector);

//echo $result_selector;
if ($result_selector->num_rows > 0) {
    //output data of each row
    while($row = $result_selector->fetch_assoc()) {
        array_push($id, $row["id"]);
        array_push($student_id, $row["student_id"]);
        array_push($class_id, $row["classes_id"]);

        //echo "id: " . $row["id"]. " - First Name: " . $row["first_name"]. "- Last name" .
$row["second_name"]. "- iam" . $row["iam"]. "- schoolyear". $row["schoolyear"]. "<br>";
    }
} else {
    echo "0 results";
}

$student_id_txt = "";

foreach ($student_id as $key => $value) {

    $student_id_txt = $student_id_txt . " " . $value . ",";
    # code...
}
//echo "<br>";
$student_id_txt = substr_replace($student_id_txt, "", -1);

//echo $student_id_txt;
$sql_student = "SELECT id, first_name, second_name, iam, schoolyear FROM students WHERE id IN ("
. $student_id_txt . ")";
//echo "<br>";
//echo $sql_student;
$result_student = $mysqli->query($sql_student);

$id_student = array();
$first_name_student = array();
$second_name_student = array();
$iam_student = array();
$schoolyear_student = array();

if ($result_student->num_rows > 0) {
    //output data of each row
    while($row = $result_student->fetch_assoc()) {
        array_push($id_student, $row["id"]);
        array_push($first_name_student, $row["first_name"]);
        array_push($second_name_student, $row["second_name"]);
        array_push($iam_student, $row["iam"]);
        array_push($schoolyear_student, $row["schoolyear"]);

        //echo "id: " . $row["id"]. " - First Name: " . $row["first_name"]. "- Last name" .
$row["second_name"]. "- iam" . $row["iam"]. "- schoolyear". $row["schoolyear"]. "<br>";
    }
} else {
    echo "0 results";
}
//echo $first_name_student[0];

//TODO input rooms, and entreprises
$sql_student = "SELECT id, first_name, second_name, iam, schoolyear FROM students WHERE id IN ("
. $student_id_txt . ")";

$result_student = $mysqli->query($sql_student);

$id_student = array();
$first_name_student = array();

```

```

$second_name_student = array();
$iam_student = array();
$schoolyear_student = array();

if ($result_student->num_rows > 0) {
    //output data of each row
    while($row = $result_student->fetch_assoc()) {
        array_push($id_student, $row["id"]);
        array_push($first_name_student, $row["first_name"]);
        array_push($second_name_student, $row["second_name"]);
        array_push($iam_student, $row["iam"]);
        array_push($schoolyear_student, $row["schoolyear"]);

        //echo "id: " . $row["id"]. " - First Name: " . $row["first_name"]. " - Last name" .
        $row["second_name"]. " - iam" . $row["iam"]. " - schoolyear". $row["schoolyear"]. "<br>";
    }
} else {
    echo "0 results";
}

$mysqli->close();

```

Basically, the PHP is needed to import data from the database and make it accessible for further use in the HTML.

And this is the HTML part:

```

<body style="margin-bottom: 50px;">

    <div id="navbar"></div>
    <script src="/scripts/navbar.js"></script>

    <div id="body" class="container2">

        <div id="main_box" class="containerr">

            <!-- classes -->

            <table class="table h-100 w-100" style="font-size: 2rem;" id="main_box">
                //display a table where thead the head column is, the tbody all the rest.

                <thead>
                    <tr>
                        <th scope="col">#</th>
                        <th scope="col">Name</th>
                        <th scope="col">Entreprise</th>
                        <th scope="col">iam</th>
                        <th scope="col">salle</th>
                    </tr>
                </thead>
                <tbody id="inp">
                    <tr>
                        <th scope="row">1</th>
                        <td>Sergonne Lou</td>
                        <td>chelsea, lite</td>
                        <td>serlo966</td>
                        <td>B.1.11</td>
                    </tr>
                    <tr>
                        <th scope="row">2</th>
                        <td>Sven bordez</td>
                        <td>Chelsea, Dragons</td>
                        <td>borsv123</td>
                    </tr>
                </tbody>
            </table>

```

```

        <td>B.1.11</td>
    </tr>
</tr>
<tr>
    <th scope="row">3</th>
    <td>José Santos</td>
    <td>Dragons</td>
    <td>sanjo456</td>
    <td>B1.11</td>
</tr>
</tbody>
</table>
</div>
</div>
</div>
<script>

var id = <?php echo json_encode($id_student); ?>;
var first_name = <?php echo json_encode($first_name_student); ?>;
var second_name = <?php echo json_encode($second_name_student); ?>;
var iam = <?php echo json_encode($iam_student); ?>;
var schoolyear = <?php echo json_encode($schoolyear_student); ?>;

# for id, write name in main_box div
const main_box = document.getElementById("inp")
for (let I = 0; I < id.length; i++) {
    main_box.insertAdjacentHTML("afterbegin", `
        <tr>
            <th scope="row">${id[i]}</th>
            <td><a class="hide_a" href="student.php?student_id=${id[i]}">${second_name[i]}
${first_name[i]}</a></td>
            <td>to be done</td>
            <td>${iam[i]}</td>
            <td>to be done</td>
        </tr>
    `)
}
</script>

</body>

</html>

```

The code starts by defining the layout of the page using HTML tags, such as body, div, table, thead, tbody, tr, th, and td and with these tags creating a table for displaying students.

In the div with id "navbar", there is no content, but it is used to reserve a place where to insert the code. The script tag imports an external JavaScript file named "navbar.js", which places the navbar HTML in the div.

Inside the div with the id "body" and class "container2", there is another div with id "main_box" and class "containerr". This div is where the dynamically generated table where the students will be listed will be placed.

Then the code defines this dynamically generated table, with the tags thead, tr, tbody, etc. Inside there are also defined 3 example students, that are there without being inserted by the JavaScript.

Tag	Description
<code><table></code>	Defines a table
<code><th></code>	Defines a header cell in a table
<code><tr></code>	Defines a row in a table
<code><td></code>	Defines a cell in a table
<code><caption></code>	Defines a table caption
<code><colgroup></code>	Specifies a group of one or more columns in a table for formatting
<code><col></code>	Specifies column properties for each column within a <code><colgroup></code> element
<code><thead></code>	Groups the header content in a table
<code><tbody></code>	Groups the body content in a table
<code><tfoot></code>	Groups the footer content in a table

All types of headers by w3schools.

The JavaScript code starts by declaring several variables (`id`, `first_name`, `second_name`, `iam`, and `schoolyear`) that are being populated with data from the server-side using PHP. The values of these variables are later used to generate the rows of the table.

The constant value, also stated as `const`, `main_box = document.getElementById("inp")` statement retrieves the `tbody` element of the table with `id "inp"`, where the rows will be inserted.

The `for` loop after the `main_box` iterates over the data arrays, imported by the variables higher up, and uses the `insertAdjacentHTML` method to dynamically create and insert new rows into the table.

Each row has five cells: the first cell displays the student ID, the second cell displays the student's name as a hyperlink that takes the user to another page showing the student's information, the third cell displays the enterprises of the student, the fourth cell displays the student's IAM, and the fifth cell displays the name of the room they are currently in.

Finally, the code imports jQuery and Bootstrap JavaScript libraries from external sources to provide additional functionality for the page, such as animations and styling.

Procedure of coding

Basically, I learned all of this while coding and making the first prototype. So, I began planning a page, thinking of what features it should include and which connections to I need, do I need a database connection (most of the times yes), do I need to get attributes or values from the page before a user get on the page, etc.

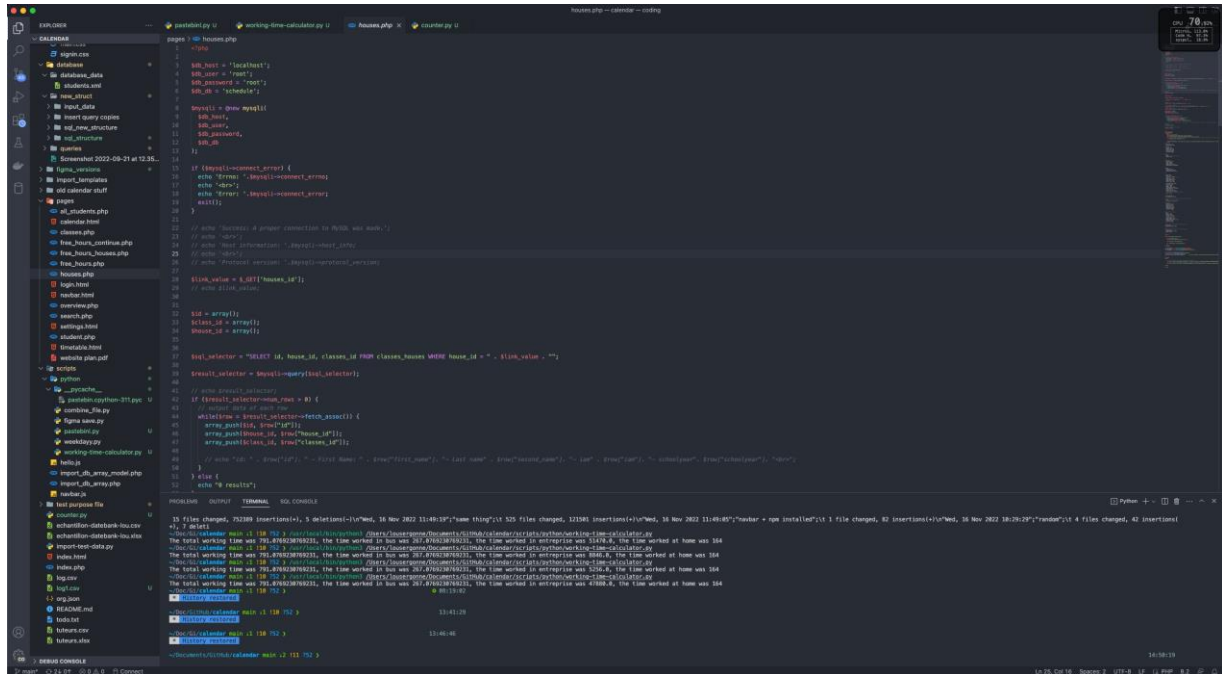
If this is sorted out, I begin to build the structure of the code, with structure I mean the PHP and HTML „introductions “, for example to initiate a PHP part in the code there has to be a „<?php“ and to end it „?>“.

Then I import the navbar with my neat little script and import all other things like CSS, fonts, and other scripts.

After these steps I finally start to code.

Depending how much features the page includes the time of building up the page varies heavily from each other, the first page I created, that was the page where you can select students to view their information’s. On this page I spent around 5-6 hours creating it so that it works, 2-3 more hours to refine and optimize the page and the code. After this I moved on to other pages, but I went back often to include features like the imported navbar code.

As code editor I used VSCode to program and to write my code, with a lot of neat features like syntax highlighting, multiple line selection, themes etc. and it being free it was easy to decide to use it. The massive amounts of plugins assisted in helping with features like the possibility to run SQL queries from out of my code browser and to have autocomplete etc.



The GUI of VSCode where I work.

To give a bit of insight how to code progressed over the time the next chapter shows the GitHub Changelog.

GitHub Changelog

The change log logs every commit I have committed to my GitHub project, and this can be interesting to view to see the phases of development in this case. In other cases, it is to log changes and rollback if something doesn't work as intended due to a change made in the code.

Here is the commit Log:

The 5 first commits

```
"Thu, 22 Sep 2022 14:43:22";"Update main.yml"; 1 file changed, 1 insertion(+), 1 deletion(-)
"Thu, 22 Sep 2022 14:35:32";"create connection to ftp deployment"; 1 file changed, 38
insertions(+)
"Wed, 21 Sep 2022 13:21:21";"same as last commit"; 4 files changed, 479 insertions(+)
"Wed, 21 Sep 2022 13:20:23";"built a liddle calendar prototype to fill in a calendar with
database input not finished yet"; 1 file changed, 71 insertions(+)
"Wed, 21 Sep 2022 10:46:28";"Create index.html"; 1 file changed, 1 insertion(+)
```

The 10 last commits

```
"Wed, 1 Mar 2023 13:15:34";"format, insert sql"; 59 files changed, 4551 insertions(+), 6
deletions(-)
"Wed, 1 Mar 2023 10:13:11";"adapt new sructure of Database"; 74 files changed, 1420
insertions(+), 1000 deletions(-)
"Wed, 25 Jan 2023 13:12:54";"continued overview.php to have the filter section"; 1 file
changed, 721 insertions(+), 6 deletions(-)
"Wed, 25 Jan 2023 10:40:29";"same thing"; 10 files changed, 414 insertions(+), 72
deletions(-)
"Wed, 25 Jan 2023 10:36:55";"same thing"; 4 files changed, 61 insertions(+)
```

And here is the whole without formatting and removing excess:

<https://pastebin.com/BDxLUFym>

As you can see here there is a lot of misspelled and brief titles, this is because of the whole purpose I used GitHub for. I used it to sync everything to a cloud. I as already mentioned use GitHub for syncing to different computers and most of the times I "saved" my work to the cloud when I was finished with my hours at my enterprise, and this had to be done quickly. Looking back, it would be cool to have in detail commits etc. as the professionals do but I cannot remember everything, so this is impossible, maybe in future projects I'll do this. Another scenario where I committed my changes, is when I started working in my enterprise and wanted to have the changes as quickly as possible to start working.

Test Hosting

This year I changed to the LITE enterprise and there they have a SFTP (Secure File Transfer Protocol) server hosted where you can put your websites and view them online. This was a great cause to explore the possibilities of DeployHQ, and I then build up a system that when I commit something in my GitHub repository, the tool would automatically update the contents of the sftp. But this approach was failing due to some errors with script execution, and I chose not to use this system due to its flaws.

Sftp

SFTP (Secure File Transfer Protocol) is a protocol for transferring files over a secure SSH (Secure Shell) connection. It allows you to securely transfer files between two machines, even across different networks or over the internet.

Here are some of the basics of SFTP:

- Authentication: SFTP uses SSH for authentication, which means that both the client and server need to authenticate each other before the transfer can take place. This is typically done using public key cryptography, where each machine has a public key and a private key. The public keys are used to encrypt data, while the private keys are used to decrypt it.
- Encryption: SFTP encrypts all data in transit using SSH's encryption algorithms, which makes it secure against eavesdropping and tampering.
- Port: SFTP typically uses port 22, which is the same port used by SSH. However, some servers may use a different port, so you should check with your server administrator to confirm the port number.
- Commands: SFTP uses a set of commands similar to those used in FTP (File Transfer Protocol), such as ls, cd, mkdir, rm, put, and get. These commands allow you to navigate the remote file system, create directories, delete files, and upload/download files.
- File transfer: SFTP supports two types of file transfer modes: binary and ASCII. Binary mode transfers files as is, while ASCII mode translates the files between different character sets. By default, SFTP uses binary mode.
- File permissions: SFTP supports file permissions, just like FTP. This means that you can set permissions on files and directories on the remote server, and control who can access and modify them.

Overall, SFTP provides a secure way to transfer files over a network, while using a familiar set of commands and file permissions.

Phase 4: Import of Data

The first dataset I used entirely for testing as I connected everything and viewed it the first time it looked empty. But because I was testing while developing the site, I needed some more example cases so, I decided to randomly generate data for testing purposes. For this I used Mockaroo, a tool to randomly create data in database format, where I entered what I wanted for each field in a database and then exported this as SQL queries, these look like this:

```
create table students (
  id INT,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  iam VARCHAR(50),
  schoolyear INT
);
insert into students (id, first_name, last_name, iam, schoolyear) values (1, 'Wanids', 'Trimby', '3wyF3196', 2021);
insert into students (id, first_name, last_name, iam, schoolyear) values (2, 'Donalt', 'Abramzon', '8RUJd491', 2022);
insert into students (id, first_name, last_name, iam, schoolyear) values (3, 'Lauralee', 'Shmyr', 'NKU7q533', 2022);
insert into students (id, first_name, last_name, iam, schoolyear) values (4, 'Nerti', 'Gumery', '4nbwx452', 2021);
insert into students (id, first_name, last_name, iam, schoolyear) values (5, 'Amye', 'Gladdolph', 'LnQgz308', 2021);
insert into students (id, first_name, last_name, iam, schoolyear) values (6, 'Madelina', 'Meeson', 'pg4Cx504', 2022);
insert into students (id, first_name, last_name, iam, schoolyear) values (7, 'Darline', 'Mateev', '9RPVe041', 2022);
insert into students (id, first_name, last_name, iam, schoolyear) values (8, 'Bambi', 'Ingley', 'xNhGp258', 2022);
insert into students (id, first_name, last_name, iam, schoolyear) values (9, 'Avivah', 'Heliet', 'zq0fu201', 2022);
insert into students (id, first_name, last_name, iam, schoolyear) values (10, 'Frank', 'Hudson', 'SP7UG686', 2021);
insert into students (id, first_name, last_name, iam, schoolyear) values (11, 'Darb', 'Hayhoe', '21KpW076', 2022);
```

And here is the whole file:

<https://pastebin.com/uD7rjXLD>

I ran the queries and tested them. This was necessary to test performance and debugging, because if it works for one entry that doesn't mean that it works for 500 entries. One example of these bugs is in the part of the website where you can select a class to view data. There I discovered that the script I used was very inefficient and caused waiting times, to display all the students, which were in the test data 500, my computer took 4-5 seconds, which is too long as a loading time for such an „little“ number. Bear in mind my computer is powerful and the computers in school would take longer as they are older and less performant. So, this was a thing I had to optimize, although I haven't found an optimal solution and the script has to still iterate over every student.

This step was important due to these little bugs and performance issues and to resolve them is also important, but I'm sure that I haven't removed every bug in the code.

Inserting real life data

The data I used previously was incomplete and made no sense, the houses were states, the classes had country names etc., and the students weren't linked specifically to a house, these connections were randomly generated and included some duplicates which caused some problems with primary and foreign keys as I implemented them. To test out the database at its full potential I then decided to take real data. I asked the local Informatician from the LEM, to receive the data from my class, and formatted it correctly to insert it into the database.

At this point I stopped working on the coding etc. and started writing on this document. I plan on continuing after the TraPe to finish the project until the next school year.

The data was in the following format as csv, and I wrote a script to take that and insert it into SQL statements to insert it in the database:

Here you can view the script:

```
import os

with open("/Users/lousergonne/Documents/GitHub/calendar/tuteurs.csv", "r") as f:
    frtut = f.read()

with open("/Users/lousergonne/Documents/GitHub/calendar/echantillon-datebank-lou.csv", "r") as f:
    fr = f.read()

frtut = frtut.split("\n")
fr = fr.split("\n")

print(frtut)
print(fr)
numtut = 0
for tut in frtut:
    if "\u0000" in tut:
        tut = tut.replace("\u0000", "")
    tut = tut.split(";")
    query_tuteur = f"INSERT INTO `tuteur` (`id`, `first_name`, `second_name`, `house`,
`entreprise`, `untis_id`, `year`) VALUES ({numtut}, '{tut[-1]}', '{tut[-2]}', 1, 1, '{tut[0]}',
2)"
    print(query_tuteur)
    final_query.append(query_tuteur)
    numtut += 1

final_query = []

numstudent = 0

for student in fr:
    student = student.split(";")
    numstudent += 1
    pass
```

Here can you view a line of the csv file, which means comma-separated values:

```
SerLo966;Sergonne;Lou;4C6;BARAL
```

I can't show the whole csv due to privacy concerns and therefore I show only my line in the csv.

Phase 5; Testing

The testing phase was not reached due to the time limitations, and I will be doing the testing afterwards I finished developing. But I already tested some things while coding. One simple test is the test if it even works, then if it works on all the standard browsers etc. Another simple measure would be testing it on other system versions, but this can also be done afterwards.

Steps of testing

To conduct these tests, one can use various testing tools and techniques such as manual testing, automated testing, load testing tools, security testing tools, and so on. Additionally, you can also use testing frameworks such as PHPUnit for testing PHP code and Selenium for testing web applications.

Functional testing

This involves testing the various functionalities of the web application such as user registration, login, data entry, search, and so on.

Database testing

This involves testing the various aspects of the MySQL database such as data integrity, data validation, data manipulation, and data retrieval.

Performance testing

This involves testing the performance of the web application and the database by simulating a high load on the system and measuring its response time, throughput, and scalability.

Security testing

This involves testing the web application and the database for potential security vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

Usability testing

This involves testing the user interface and user experience of the web application to ensure that it is easy to use and meets the needs of the users.

Compatibility testing

This involves testing the web application and the database on various web browsers and devices to ensure that it is compatible with different platforms and configurations.

Phase 6: Deployment

The deployment phase never happened, because when the time came to submit my TraPe, I wasn't nearly finished and only have a prototype to show off. Hopefully I will have finished the project until the next schoolyear so that I can deploy it, even if it is only for one enterprise.

To have something to show off I hosted an incomplete version on a mini server that I host on my own and everyone can access it through this link:

<https://unkreative.github.io/calendar/>

https://unkreative.github.io/calendar/pages/all_students.html

This version of the site does not work fully. You can view the index site, and then click on all students, and then there explore, but the rest is not supported and will give you error messages. This is due to github pages being a static hosting service and PHP is dynamic.

But hypothetically a deployment of a web application with a MySQL database involves some steps.

I would have to

1. Choose a hosting provider.
2. Configure the server.
3. Upload the web application files.
4. Import the database schema and data.
5. Test the web application.

Problems while working

General Difficulties

In general time was a constraint, because I am a one-man-team, also known as full stack developer, I had a lot to do, from design, to backend, to frontend and Server Administrator. I knew that the project would be a challenge and as I proceeded to work on it, I realised that I could not possibly finish it even though I had a lot of time to work on it.

Time constraints.

This topic has been on my mind throughout the entire project, and I've often wondered if I could complete it successfully.

Working time calculations

To showcase when I work, I did some calculations to get my estimated work amount. My actual working times are irregular and can therefore only be estimated but I tried my best estimating.

Each Day I take my bus where I can effectively work 45-55 minutes, this time interval I will call *btime*.

Subtract the times I don't take the bus, that is two times a week and I do not work every time on my project when I am in the bus, I estimate that I work around 65% of times on my project. So *bweektime* (bus work time per week) is *btime* times 2 times 5 minus 2 times *btime*, and of this take 65%. To visualize this here is the equation:

$$bweektime = (btime * 2 * 5 - btime * 2) * \frac{1}{65}$$

So, this times the school weeks minus 5 (because I was in a stage and in the beginning of the year, I didn't work on it) is the total time that I invested only in the bus.

Here is the final equation for the bustime calculation.

$$byeartime = bweektime * weeks$$

But this isn't the only place where I work on my projects. I also do in school.

I have 8 plages of enterprise, 4 in Chelsea 4 in Lite, where I can work on my project. In Chelsea I worked the whole year on it, in Lite only about half the time because of other projects.

So, this ends up in this equation =

$$eweektime = 8 * 45 \text{ minutes}$$

And for the whole year =

$$eyeartime = \frac{eweektime * weeks}{0.25} - eweektime * 5 - 10$$

The last bit of equation is to calculate in the times I worked on other things or were sick.

I also worked from home a lot; there I can only set a vague number because that differs a lot. But I estimated that I at least work 4 hours per week outside of school and bus. As already described, that depends on when and fluctuated a lot, but this is a reasonable estimate.

$$hweektime = 4$$

and for the year that makes =

$$hyeartime = hweektime * weeks + hweektime * 4$$

The last bit of the equation was for holidays where I worked more than during school time.

So, all this combined creates this equation =

$$totaltime = byeartime + eyeartime + hyeartime$$

Here is the equation in form of python code because I didn't want to calculate it by hand (the time of weeks was also estimated).

```
weeks = 37
btime = 70
# time spent in week
bweektime = (btime * 2 * 5 - btime * 2)/65
byeartime = bweektime * weeks - bweektime * 6
```

```

eweektime = 8 * 45
eyeartime = (eweektime * weeks)/0.25 - eweektime * 5 - 10
hweektime = 4
hyeartime = hweektime * weeks + hweektime * 4
total = byeartime + eweektime + hyeartime

print(f"The total working time was {total}, the time worked in bus was {byeartime}, the time worked in enterprise was {eyeartime}, the time worked at home was {hyeartime}")

```

So, this python code spitted out these numbers.

“The total working time was 791.0769230769231, the time worked in bus was 267.0769230769231, the time worked in enterprise was 51470.0, the time worked at home was 164”.

This estimate seems high but could be true because I did not calculate in the time I worked in the etudes and in the rest of the schooltime.

Time constraint

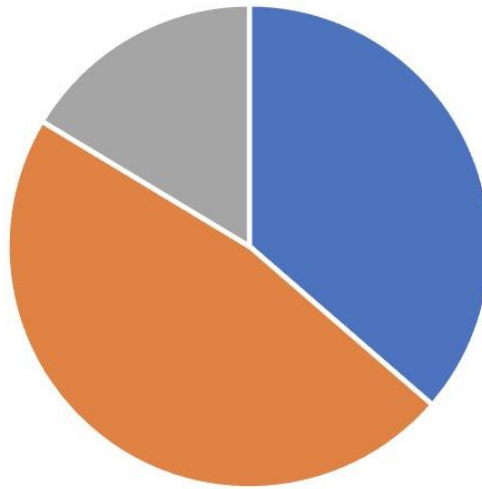
As time in a year is finite and I could not work the whole time on this. If I would be working 8 hours a day on the project I may be finished by now, but because of other stuff this isn't possible. Then there is the added difficulty of a Full stack developer (A full stack developer is a software engineer who has the ability to work on both the front-end and back-end aspects of a web application) + Designer which is directly linked with the time one needs to complete a project. For example, a whole Team of designers and developers would be far faster than me alone.

This is one of the reasons I could not finish the project in time.

Learning part

As we viewed in the chapter earlier, the estimated time I worked on my project is about 791, this time can be divided in 3 parts, learning time, working, and planning and thinking. Here's a pie chart to show the proportions of the 3 categories:

Time calculations



The grey part is the thinking part, the blue one is the learning part, and the orange part is the coding part or working part.

Learning takes up a whole lot because I started to program only 2 years prior, and the form of web development was new to me. So meanwhile working on it I learned a lot of things and how they are done. Looking back on this I don't regret it; the one part is knowing how to do the stuff but what I also learned and improved is the part of learning how to learn and to solve problems.

At first my approach to learning a new programming language was straightforward. Watch a YouTube tutorial, recreate what is showed, mostly by copying the code and then check a checkbox that I learned this. This method is good to get to know a language but to achieve or reach "higher" levels of the language I figured that is more effective to learn on the go while working on a project.

One example of this is PHP, I never looked at any tutorial on YouTube or a step-by-step guide to accomplish a whole task. I rather stuck to the documentation and a good friend of every programmer, StackOverflow (a forum where people ask questions related to coding) and had a lot more trial and error situations where I played around in the code and through this process, I learned in my opinion more as I would from the other method.

Tough this is only a personal preference and both ways work, and I'm certain that there are other ways.

But one thing I noticed that I learned to learn "faster" in some sort of way, I have kind of trained a filter where I recognise a lot faster where the important part is that I need.

Definitions

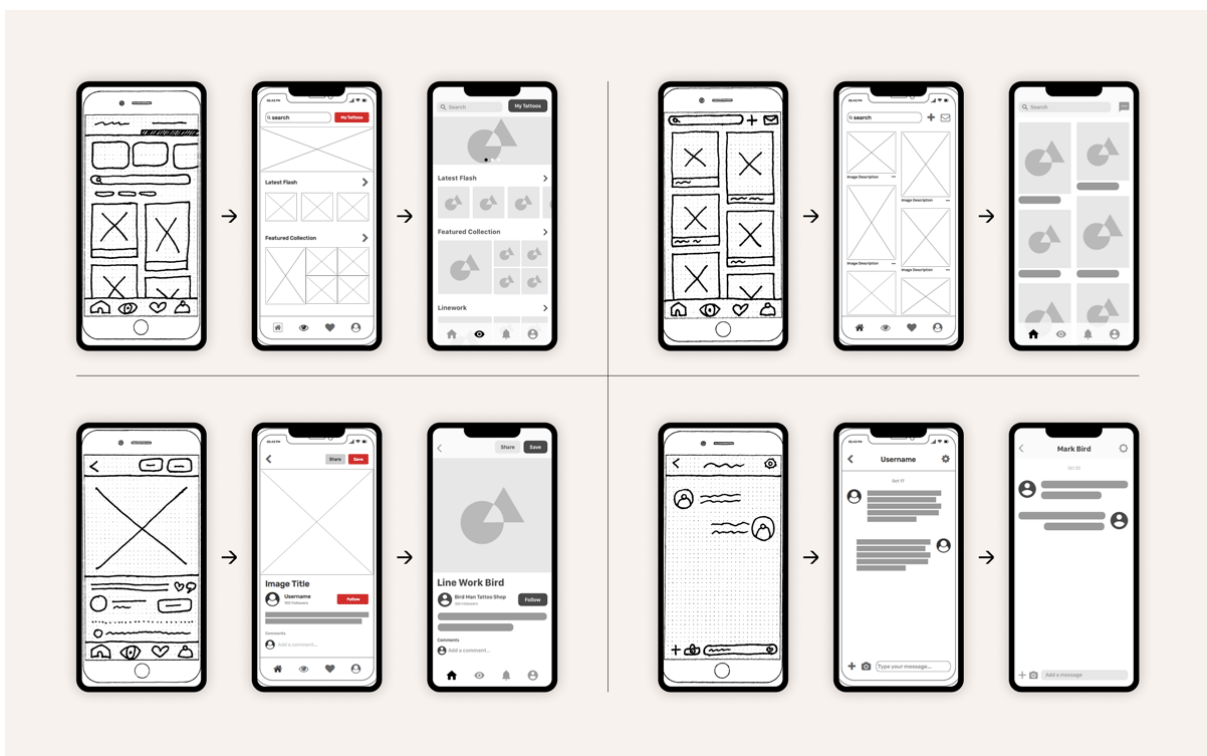
This chapter acts as a little dictionary for explanations of terms and concepts.

Design

Wireframes

A website wireframe is a low-fidelity representation of the final design, using simple shapes and placeholders to represent content and features.

Website wireframes are used in the early stages of web design to help visualize the layout and structure of a website and test different design concepts and gather feedback from users.



The progress of creating a wireframe

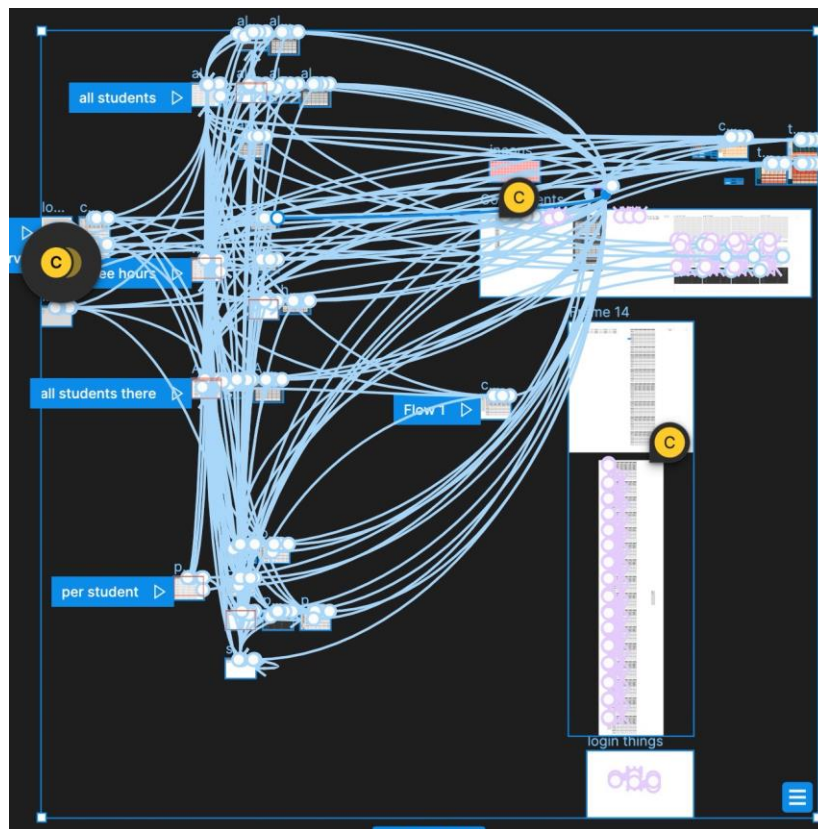
Wireframes include a layout grid, placeholders for content, and notes or annotations to explain the purpose of each element on the page. They ensure the final design meets the needs of users and stakeholders and is easy to use and navigate.

Figma

Figma is a popular design tool used by designers to create user interfaces, wireframes, prototypes, and designs for digital products such as websites and mobile apps. Its primary objective is to provide designers with a comprehensive set of tools that can streamline the design process and improve collaboration with team members and stakeholders.

With Figma, designers can edit vectors, create interactive prototypes with animations and transitions, collaborate in real-time with stakeholders, keep track of design versions, and export design assets and specifications for developers to use when building the product.

A feature important for my use case is the linking feature enabling me to create buttons and the page connected to it.



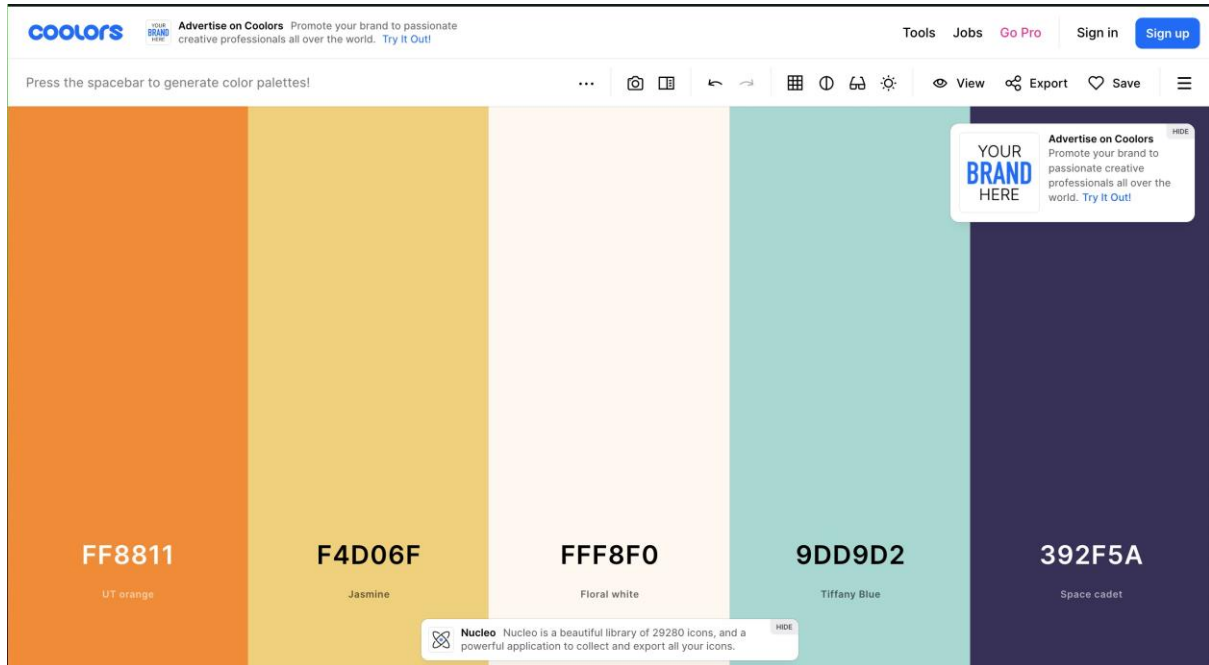
Here are all connections in my final Figma File, every blue and pink line represents such a link.

By offering these features, Figma helps designers to create high-quality digital products more efficiently, saving time and ensuring everyone involved in the project is on the same page.

Overall, Figma is a valuable design tool that empowers designers to create better products and help organizing the process of creating a website or Application.

Coolors

Coolors is a website that allows you to generate colour schemes for various design projects such as websites, graphic designs, and more. It provides an easy way to explore, discover or pull palettes from a photo or image, and even lists trending palettes. Additionally, the website provides a fun interface where you can test the generated colour schemes, adjust the colours, and save your favourite palettes for later use.



Here is their colour generator.

And if you want you can explore if for yourself:

<https://coolors.co/generate>

Databases

Databases in General

A database is a structured set of data that is stored and organized in a way that allows for efficient retrieval and manipulation of that data. Databases are used in many applications, including websites, mobile apps, and business software. They can be classified into different types, such as relational databases (such as MySQL, PostgreSQL), NoSQL databases (such as MongoDB, Cassandra), and graph databases (such as Neo4j). Each type has its own strengths and weaknesses, and choosing the right type depends on the specific needs of the application.

SQL

Structured Query Language (SQL) is a programming language used for managing and manipulating data in relational database management systems (RDBMS) or in simple terms, to manage a database. It is commonly used in the development of various software systems, including web-based applications, enterprise software, and other database-driven systems.

The purpose of SQL is to perform a range of tasks such as querying data, inserting new data, updating existing data, and deleting data from a database. It uses a syntax that resembles natural language and includes statements such as SELECT, INSERT, UPDATE, and DELETE, as well as clauses such as WHERE, ORDER BY, GROUP BY, and JOIN, that refine and filter the results of queries.

An example of a SQL query is:

```
INSERT INTO `Tuteur` (`id`, `first_name`, `second_name`, `house`, `enterprise`, `untis_id`, `year`) VALUES (NULL, 'Sophie', 'THOMA', 1, 1, 'THOSO', 2)
```

This query inserts the values id, first_name, second_name, house, enterprise, untis_id and year into the table Tuteur. The values are defined in the parentheses after the Word VALUES, where id equals NULL, first_name equals Sophie, second_name equals THOMA, etc.

SQL is a powerful tool for managing and manipulating large amounts of data, and it is used by many organizations across industries to store, analyse, and report on data.

However, using SQL requires careful design and management to ensure data security, scalability, and performance. Therefore, I remade my Database multiple times because it was built to be inefficient.

MySQL

MySQL is an open-source relational database management system. It is used to store, organize, and manage data. MySQL uses a client-server model, where a client communicates with the server to access the database. It is widely used in web applications to store user data, such as login credentials and preferences. MySQL supports multiple storage engines, allowing you to choose the most appropriate one for your application. It also supports transactions, which ensure that multiple changes to the database are performed as a single, atomic operation.

PhpMyAdmin

PhpMyAdmin is a free and open-source web-based tool written in PHP for managing MySQL databases. It provides a user-friendly interface for performing various tasks such as creating, modifying, and deleting databases, tables, and fields, executing SQL queries, and managing user permissions. PhpMyAdmin is widely used by web developers and webmasters to manage their MySQL databases, especially when they are not familiar with command-line tools. It allows users to easily interact with their databases without needing to learn complex SQL commands.

Foreign keys

A foreign key is a field in a database table that references (mention or link) the primary key of another table. It is used to establish a relationship between two tables in a database. The foreign key ensures that the data stored in the referencing table is consistent with the data stored in the referenced table. When a record is added or updated in the referencing table, the foreign key constraint requires that the corresponding record exists in the referenced table, or the operation will fail. This ensures data integrity and prevents the creation of orphaned records in the database. In simple terms, a foreign key is a way of linking two tables together and ensuring that they remain synchronized.

Primary keys

A primary key is a unique identifier for a value in a database table. It is a column that uniquely identify each row in a table. The primary key ensures that each value in a row is uniquely identifiable and can be accessed quickly and efficiently. It also serves as a reference point for other tables that may reference the primary key as a foreign key.

In most relational databases, the primary key is implemented as an index, which allows for faster searching and sorting of the data. In my case the index is also the primary key and is also used as reference point for foreign keys.

Overall, the primary key is an essential part of any database table as it provides a reliable way to uniquely identify each record in the table and maintain the integrity of the data, which is very important for a good working program environment.

Unique fields

A unique key in a database is a constraint that ensures that the values in a column in a table are unique and cannot be duplicated. The unique key is similar to a primary key, but it differs in that it does not serve as the primary identifier for a record in the table. Instead, it provides another level of data integrity by ensuring that no two rows in the table can have the same values in the unique key column(s), in certain cases that would be counterproductive, for example there are no duplicate IAM identifiers.

Another feature in common with primary keys, unique keys are implemented as indexes. They can also be used as foreign keys to link tables together.

Indexing tables

In a database, an index is a data structure that allows for faster searching and sorting of data in a table. It is created on one column in a table and provides a quick way to locate specific data based on the values in those columns.

The most common type of index is a B-tree index, which stores the index data in a balanced tree structure. This allows for efficient searching of the data because the search algorithm can quickly navigate the tree to find the desired data.

However, B-tree indexes takes up a significant amount of memory, especially in large tables with many columns. This is because the index must be maintained and updated every time the data in the table changes. As a result, creating too many indexes on a table can slow down insert and update operations. Therefore, you should not assign every row an index and it should be used only if needed.

One to one, many to many, etc

In a database that has relational data there are some ways to reference data, one way we already viewed are Primary keys and foreign keys, but if many items have many attributes or on item has a lot of attributes this system doesn't work anymore and there is a need for a linking database.

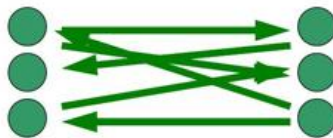
One to one relationships connect one entity to one other entity:



One to many relationships connect one entity to one or more other entities:



Many to many relationships connect many entities to many other entities:



Here is a visual explanation for these types of connections.

So, to connect one to many, there must be a connecting database. These connecting databases could look like this:

```
id student_id entreprise_id room plage_1 plage_2 plage_3 plage_4 plage_5 plage_6 plage_7 plage_8 year
```

Where `student_id`, defines the student, and `entreprise_id` defines the enterprise the student is in. So, with the combination of these ids, you can link data together. Another purpose this table fulfils are the information when each student has which plage.

Mockaroo

Mockaroo is a website that allows users to generate realistic test data for their applications. With Mockaroo, users can create custom data sets with various data types such as names, addresses, phone numbers, and more. The website has a user-friendly interface that enables users to set up their data sets quickly and easily. It also offers features such as data validation and the ability to download the data in various formats. Overall, Mockaroo is a powerful tool for developers and testers looking to test their applications with realistic data.

<input type="checkbox"/>	Classes id name schoolyear	5 months ago	
<input type="checkbox"/>	classes_free_hours id class_id free_hour1 freehour2	3 months ago	
<input type="checkbox"/>	classes_houses id classes_id house_id schoolyear	5 months ago	
<input type="checkbox"/>	entreprise id name CEO schoolyear	5 months ago	
<input type="checkbox"/>	entreprise_staff id staff_id entreprise_id post schoolyear	5 months ago	
<input type="checkbox"/>	houses id name house_number schoolyear	5 months ago	
<input type="checkbox"/>	house_staff id house_id staff_id function schoolyear	5 months ago	
<input type="checkbox"/>	staff id first_name last_name schoolyear	5 months ago	
<input type="checkbox"/>	student_entreprise id student_id entreprise_id plage_1 plage_2 plage_3 plage_4 plage_5 plage_6 plage_7 plage_8	5 months ago	
<input type="checkbox"/>	students id first_name last_name iam schoolyear	5 months ago	
<input type="checkbox"/>	students_classes id student_id classes_id schoolyear	5 months ago	
<input type="checkbox"/>	timetable id interval schoolyear	5 months ago	
<input type="checkbox"/>	tuteur id first_name second_name house entreprise	5 months ago	
<input type="checkbox"/>	tuteur_student id student_id tuteur_id schoolyear	5 months ago	

Here is a list of all tables I have generated random data for.

Field Name	Type	Options
id	Row Number	blank: 0%
name	Country	restrict countries... blank: 0%
schoolyear	Number	min: 2021 max: 2022 decimals: 0 blank: 0%
+ ADD ANOTHER FIELD		GENERATE FIELDS USING AI...
# Rows:	50	Format: SQL Table Name: classes <input type="checkbox"/> include CREATE TABLE
Append Dataset:	choose a dataset...	
Generate data using cURL with the following command:		
<pre>curl "https://api.mockaroo.com/api/216a9670?count=50&key=29da6730" > "Classes.sql"</pre>		
Public URL: https://www.mockaroo.com/216a9670		

And here is a picture of the GUI

Coding

Python

Python is a high-level, interpreted programming language that is widely used for web development, scientific computing, data analysis, artificial intelligence, and many other applications. It was created in the late 1980s by Guido van Rossum and has since become one of the most popular programming languages in the world. Python is known for its simplicity, readability, and ease of use. It has a vast standard library and a large community of developers who contribute to its ecosystem with packages and frameworks that enhance its functionality. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Web

HTML

HTML (Hypertext Markup Language) is the standard markup language used for creating web pages and web applications. It provides the structure and content of a web page, defining the various elements such as headings, paragraphs, images, and links.

Some key features of HTML include:

- Tags: HTML uses tags to define elements on a web page. Tags are enclosed in angle brackets (< >) and are used to specify the beginning and end of an element. For example, the <p> tag is used to define a paragraph element, while the tag is used to insert an image.
- Attributes: HTML tags can also include attributes, which provide additional information about an element. For example, the tag can include attributes such as "src" to specify the URL of the image and "alt" to provide alternative text in case the image cannot be displayed.
- Nesting: HTML elements can be nested inside each other, allowing for more complex structures. For example, a (unordered list) element can contain multiple (list item) elements.
- Semantic markup: HTML provides a set of semantic elements that help to structure the content of a web page in a meaningful way. For example, the <header> element is used to define the header of a web page, while the <nav> element is used to define the navigation section.

HTML is a critical tool for creating web pages and web applications, and it provides the foundation upon which the visual design and interactivity of a website are built.

JavaScript

JavaScript is a programming language that is primarily used for creating interactive web pages and web applications. It was created in 1995 by Brendan Eich and has since become one of the most popular programming languages in the world, being used both on the front-end and back-end of web development.

Some of the key features of JavaScript include:

- **Dynamic behaviour:** JavaScript allows web developers to add dynamic behaviour to web pages, such as responding to user input, animating page elements, and modifying the content of the page based on certain conditions.
- **Interactivity:** JavaScript enables developers to create interactive user interfaces, such as dropdown menus, pop-up dialogs, and more.
- **Versatility:** JavaScript is a versatile language that can be used for a wide range of tasks, including creating web games, building web applications, and even creating server-side applications.
- **Cross-platform compatibility:** JavaScript code can be executed on a wide range of platforms, including web browsers, servers, and mobile devices.
- **Large developer community:** JavaScript has a large and active developer community, which means that there are many resources, libraries, and frameworks available for web developers to use.

JavaScript is another critical tool for modern web development, and it plays a crucial role in creating interactive and engaging web applications.

PHP

PHP is a scripting language used for creating dynamic web pages and applications on the server-side. It was developed in 1994 by Rasmus Lerdorf and is now maintained by a vast community of developers. HTML code can easily integrate PHP, allowing developers to generate dynamic content, interact with databases, and perform other tasks. PHP has several features, including compatibility with various operating systems and web servers, built-in support for databases, a broad range of functions, support for object-oriented programming, and easy integration with other web technologies. PHP is also known for its user-friendly interface and low learning curve.

CSS

CSS stands for Cascading Style Sheets, and it is a language used to describe the presentation of HTML and XML documents, including colours, layouts, fonts, and other visual elements.

In simple terms, HTML provides the structure of a web page, while CSS provides the styling and visual design. This separation allows developers to easily make changes to the visual design of a website without having to modify the underlying HTML code.

Some of the key features of CSS include:

- A variety of selectors for targeting specific HTML elements, such as tags, and IDs.
- A wide range of properties for defining the visual appearance of elements, such as colour, font size, spacing, and positioning.
- The ability to use CSS media queries to create responsive designs that adjust to different screen sizes and devices.
- Support for advanced layout techniques, such as flexbox and grid layouts.

CSS is used by web developers to create visually appealing and responsive websites, and it is an important tool for creating modern, professional-looking web pages.

Git

Git is a distributed version control system that is primarily used for software development. It was created by Linus Torvalds in 2005 to help manage the development of the Linux kernel, but has since become a popular tool for managing code in many different types of software projects.

Git allows developers to keep track of changes to their code over time, collaborate with others on code, and easily revert to previous versions of their code if necessary. Some of the key features of Git include:

- Distributed version control: Git is a distributed system, which means that every developer has their own copy of the code repository. This makes it easy for developers to work on code in parallel, and to merge changes made by multiple developers into a single repository.
- Branching and merging: Git makes it easy to create branches of code for experimentation and development, and to merge changes from different branches into a single codebase.
- Commit history: Git provides a detailed history of changes made to the code repository, including who made the changes, when they were made, and what changes were made.
- Collaboration: Git makes it easy for developers to collaborate on code, including reviewing and commenting on each other's code, suggesting changes, and merging changes into the codebase.

Git is an essential tool for modern software development, and it is widely used by developers and organizations around the world.

GitHub

GitHub is a web-based platform that provides hosting for software development and version control using Git. It allows developers to store and collaborate on code, track changes to code over time, and manage software projects.

GitHub offers a range of features and tools for software development, including issue tracking, project management, code review, and more. It also provides a social network-like experience, allowing developers to follow and interact with each other, share code, and contribute to open-source projects.

GitHub has become one of the most popular tools for software development, and it is widely used by developers and organizations around the world.

VSCode

Visual Studio Code (VS Code) is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is highly extensible and customizable, allowing users to install a wide variety of plugins to enhance their development experience. It is popular among developers due to its ease of use and versatility.

Pastebin

Pastebin is a web-based platform that allows users to store and share text online. It was created as a tool for programmers to share code snippets, but it is now used by a wide range of users for a variety of purposes.

When you use Pastebin, you can enter any text you want into a text box and then click a button to generate a unique URL for that text. You can then share that URL with other users, who can access the text by visiting the URL.

GUI

A GUI is a type of user interface that uses graphical elements such as buttons, icons, menus, and windows to make it easier for users to interact with software applications.

Conclusion

To sum up my work, I have no regrets about selecting this year's TraPe and it has been a truly enjoyable experience, filled with plenty of fun and learning.

From the beginning of my project, there's been significant progress, as evidenced by the 10,000 words of code. As I worked on the project, I acquired a wealth of knowledge in PHP. This has increased my confidence in using the language, allowing me to utilize it more efficiently.

Exploring front-end programming languages such as CSS, HTML, and JavaScript was an interesting and educational experience. However, in today's professional world, prewritten packages, and frameworks such as NPM and React are more common.

During my work, I also discovered many useful tools that can be applied in other situations. One such tool is PhpMyAdmin, which I found accidentally while searching for a localhost server. This led me to MAMP and its sister software, XAMPP, both of which can be helpful in future projects. I decided to use PhpMyAdmin to manage my databases.

Another tool I came across is Figma, which helped me for creating website wireframes. Figma will be valuable for me in my future projects for designing, co-working, and communicating with clients or team members.

As previously mentioned, working on the project was an enjoyable experience, that confirmed my passion for coding and motivated me to pursue other coding challenges, such as the advent of code. This event consists of 24 mini programming puzzles that require solving with code, as shown in the Pastebin code snippet below.

<https://pastebin.com/vWYksgYY>

And here is the piece of code that I used to solve it.

```
inp = inp.split("\n\n")
results = []

for item in inp:
    item = item.split("\n")
    cache = 0
    for number in item:
        cache += int(number)
    results.append(cache)

global highest_num
highest_num = 0

for total in results:
    if total > highest_num:
        highest_num = total

print("the highest amount of calories carried by an elf is:", highest_num)
```

Unfortunately, I haven't completed my scheduling website yet. That's my greatest regret. I did my best, but I discovered that a full-stack developer has more things to think off and responsibilities than I initially realized. Even dedicating 8 hours a day to the project, it's still time-consuming since this is a one-man show ;).

My goal is to finish it before the beginning of the next academic year, so I can deploy the program and see it being used in real-life situations.

To provide an idea of my project, I have created a test host that is accessible through these links:"

<https://unkreative.github.io/calendar/>

https://unkreative.github.io/calendar/pages/all_students.html

Thank you for reading my TraPe!

I hope you enjoyed it as much as I did work on it this year.

Any feedbacks are welcome.

I also want to thank all the people who contributed and helped me with this project. Your help was greatly appreciated.

Source and Inspiration

For sources there isn't much to write here because it is a practical TraPe and there are besides the sources for learning and for some information like names explanations etc.

Learning Resources

For the most part when working on my project, I simply googled and looked at the first sites that came up, and then implemented tested out the thing I just learned. To list some of the sources I use, I chose to list the sites that have often reoccurred:

- <https://www.geeksforgeeks.org/>
- <https://www.w3schools.com/>
- <https://stackoverflow.com/>
- <https://www.udemy.com/>
- <https://www.udacity.com/>
- <https://www.freecodecamp.org/>
- Etc.

Information resources

For specific information like abbreviations, I used mostly Wikipedia and the built in Google popups. So here is a list for sources I used to find some of the information like what SFTP means etc.:

- <https://www.wikipedia.org/>
- <https://www.geeksforgeeks.org/>
- Google popups
- Etc.

Human resources

While working on my project I utilized a lot of concepts or tricks, and these come often from people that I asked for some advice.

Staff from Nvision

- Their CEO helped me to organize the TraPe Structure and guided me through the whole process of my project.
- The Head of development, that helped me reorganise my Database, and to advance faster while coding.

Staff from the LEM

- Michael Osborne the Informatician, he gave me the test data and discussed with me about database clients.
- Pierre Obertin, also an Informatician that helped me creating the second try of the database.
- Alex Bara, for helping me in the designing process in Figma.



