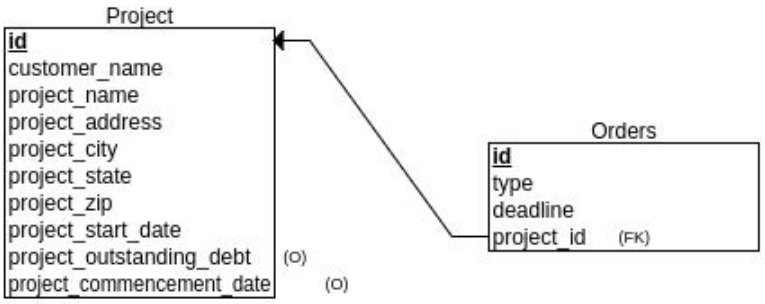
Project Management System Documentation

This is a documentation for the steps I took to implement this task

Database:

I have made a design for the database as 2 Tables one for the Projects and one for the Orders, the schema described below



This Represents a one-to-many relationship where one project has many orders or many-to-one relationship where many orders belong to a single project

I have used the doctrine documentation to create the database, the entities and define the relationship between the tables

Controllers:

Next thing was the controllers part where all the task logic layout
Along with the default controller project came with we have the ProjecrController, Which have our functions

indexAction:

Our first function which has a route to the home page and i displays to the user a form which he can upload his csv file and submit it to go to the importAction function

importAction:

- Here we take the submitted file and begin to iterate on it ignoring the first line as its just the columns names, here is an example for how the csv file should look like

	A	B	C	D	E	F	G	H	I	J	K
1	Customer Name	Project Name	Project Address	Project City	Project State	Project Zip	Project Start Date	Project Outstanding Debt	Project Commencement Date	Order Type	
2	Customer1	Project X	Project X Address	Giza	EG	11121	2018-12-01	4999.99	2019-01-01	Notice	

- Here we have from column ‘A’ to column ‘G’ are all required fields, so our function checks if those columns are not null, if any of them is null or all of them it displays a message to the user after the import process that it can’t import this row as required fields are missing

- Also it checks on the column ‘F’ as it should held an integer of 5 digits no more no less otherwise it’s an error and this row can’t be imported

- If the row has no errors and can be imported the function imports it and calls on createProject, then createOrder, and then moves to the next row

createProject:

Out of the row of data it has it checks on the columns ‘C’,’D’,’E’, and ‘F’ which uniquely identify a project

- If all of them together existed in the database then it feach that project object from the database and begin to update it the non unique columns of the project and save it again in the database
- If the unique fields don’t exist together in the database then it’s a new project and it creates and new object with the data provided then save it in the database
- Also it makes sure that if the commencement date is not provided as it’s not a required field it assign it with the ‘now’ date

createOrder:

- This one takes the project object that have been just created and assign an order to it according to column ‘J’ which helds the type of the order
- Also it assign a deadline for the order based on calculation for each type of the orders
- And if type is not supported it just tells the user that order can’t not be created as type is not supported, about the calculation those are different functions

calculateNoticeOrderDeadline:

- This function calculates the deadline for order type notice which actually relay on the project commencement date or column ‘I’, it takes the commencement date see which month is the date, take the **last day** of that month and add 60 days on it and this is the deadline
- Quick example:
  Let’s say our commencement date is 2018-02-02,
  so the date is in Feb 2018
  Last day of **Feb 2018 is 28**,
  so to get the deadline we should add 60 days on that date **2018-02-28 + 60 days**
  which gives us **2018-04-29** and that’s it

- The Notice type has special case when the city of the project is Texas, it just add 15 days to the commencement date and that the deadline

  Which means if our commencement data is **2018-02-02** then our deadline is **2018-02-02 + 15 days** which gives us **2018-02-17**

calculateLienOrederDeadline:

- This one make calculation for the other type of orders which is Lien, and the lien type this time relay on the project start date ot column 'G' it takes the project start date see which month is the date, take the **last day** of that month and add 90 days on it and this is the deadline
- Quick example:
  Let's say our start date is 2018-02-02,
  so the date is in Feb 2018
  Last day of **Feb 2018 is 28**,
  so to get the deadline we should add 90 days on that date **2018-02-28 + 90 days** which gives us **2018-05-29** and that's it

Now there is a tiny detail that we need to handle, deadlines can not be within the weekends or in the holidays, it's not acceptable, so before we return the deadline of the notice or the lien types, we need to check if the date is either a holiday or a weekend, and for this we have the next coming function

checkWeekend:

- In this function we define the weekend and holidays we have,
- First we have Saturdays and Sundays our weekends , then a snippet of the holidays (New Year's Eve 12-31,
  National Doughnut Day 06-02,
  Thanksgiving 11-24,
  Second Friday in December (example: 2016-12-09))

- Now in this function we take the deadline calculated from either of the previous 2 function and check the day of the date
  - if it is sunday we make the deadline earlier by 2 days
  - It it is saturday we make the deadline earlier by 1 day
  - If the deadline falls on a holiday we check the day before the holiday
  - If the day before the holiday is sunday then we make the deadline earlier by 3 days (sunday as weekend , saturday as a weekend, and the deadline day which is obviously an monday)
  - If the day before the holiday is saturday we make the deadline earlier by 2 days(saturday as a weekend and the holiday day which is sunday in this case)

So in our previous example on the caculateNoticeOrderDeadline we come with a deadline on 2018-04-29 which if we checked manually in the calendar we will figure out it comes on sunday , in this case the final deadline would be on **2018-04-27**

Now Out of the data we could use it to make 2 kinds of report

- First report displays Customer name, Project name , Order type and deadline of the order
- Second report for a Customer name with total number of projects, total number of order and total debt on him

projectReport And outstandingdebt Report:

This 2 functions are made basically on the queryBulider concept of symfony, I used it to write down my queries to fetch the required data and display it in tables on the interface