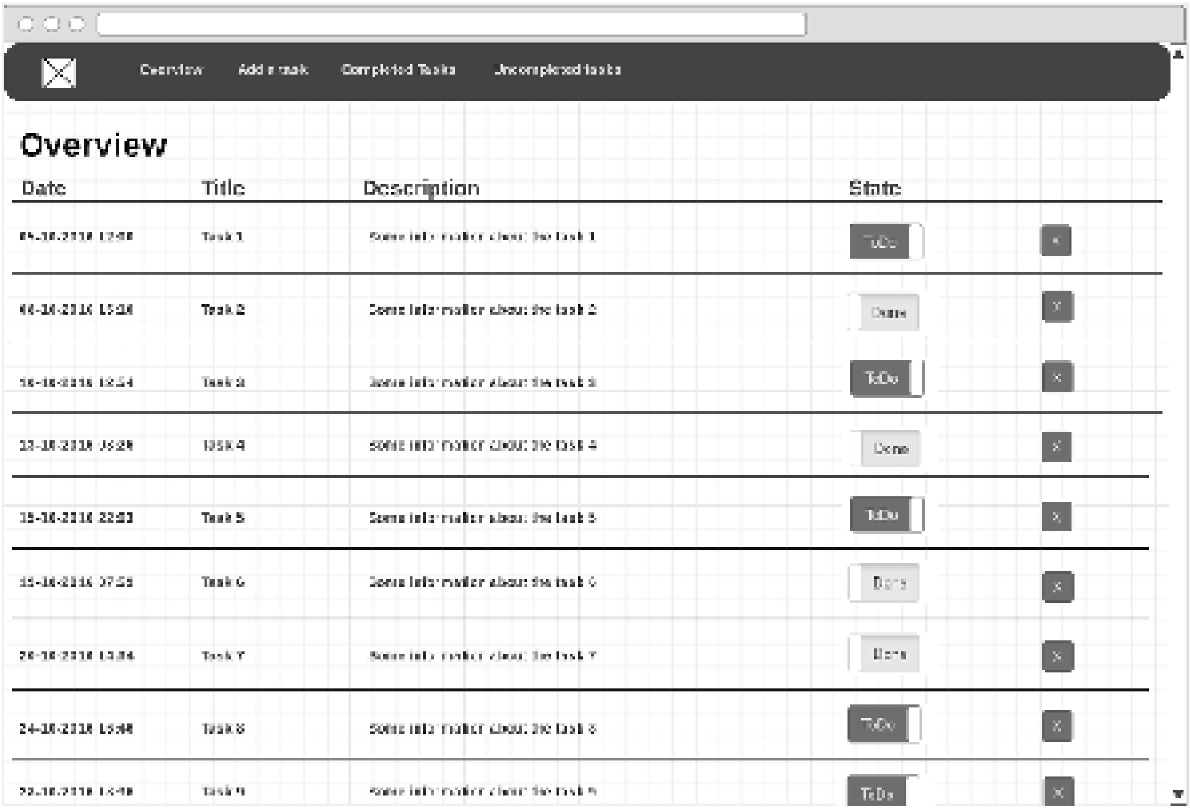
Sadeg Nabil
sc15sn
200960399

Web Application Development
Coursework 1
ToDo List

November 2016

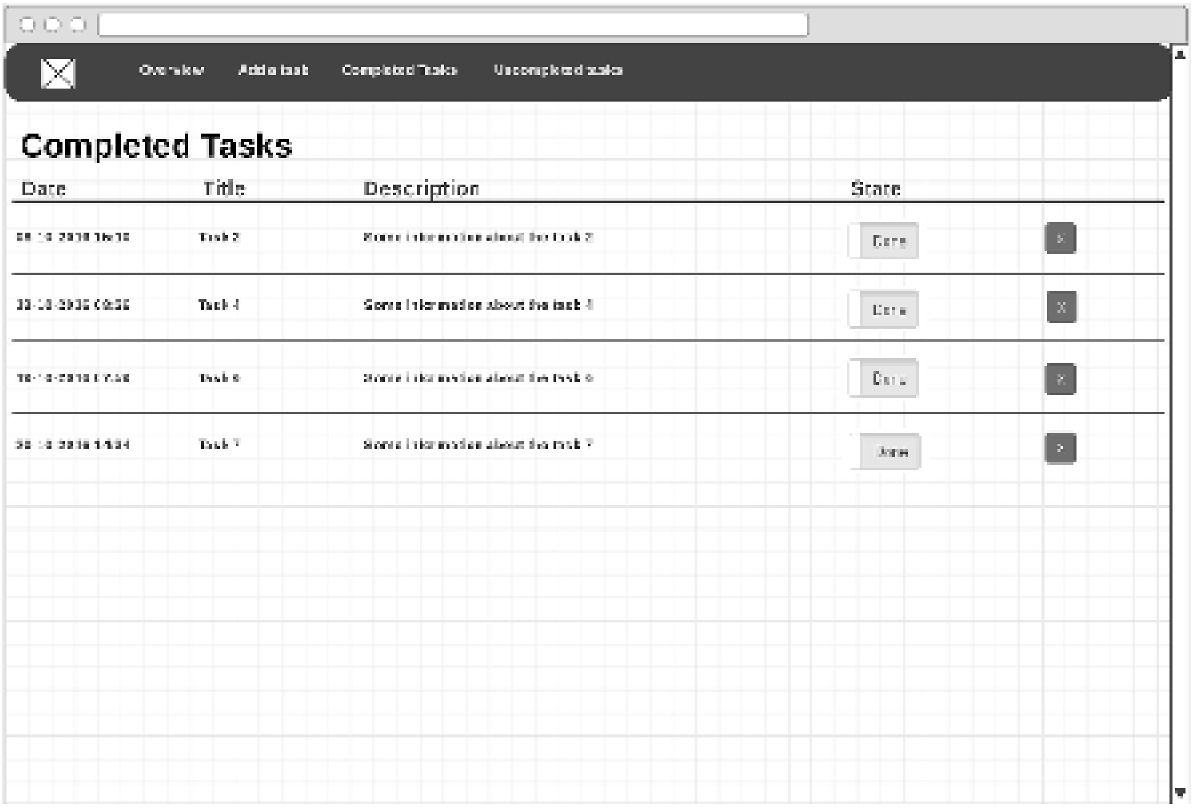
I. Wireframe



Overview

Date	Title	Description	State	
[illegible]-10-2016 12:00	Task 1	Some information about the task 1	ToDo	X
[illegible]-10-2016 15:10	Task 2	Some information about the task 2	Done	X
[illegible]-10-2016 12:54	Task 3	Some information about the task 3	ToDo	X
13-10-2016 05:20	Task 4	Some information about the task 4	Done	X
15-10-2016 22:03	Task 5	Some information about the task 5	ToDo	X
15-10-2016 07:55	Task 6	Some information about the task 6	Done	X
20-10-2016 13:04	Task 7	Some information about the task 7	Done	X
24-10-2016 15:48	Task 8	Some information about the task 8	ToDo	X
[illegible]-10-2016 13:48	Task 9	Some information about the task 9	ToDo	X

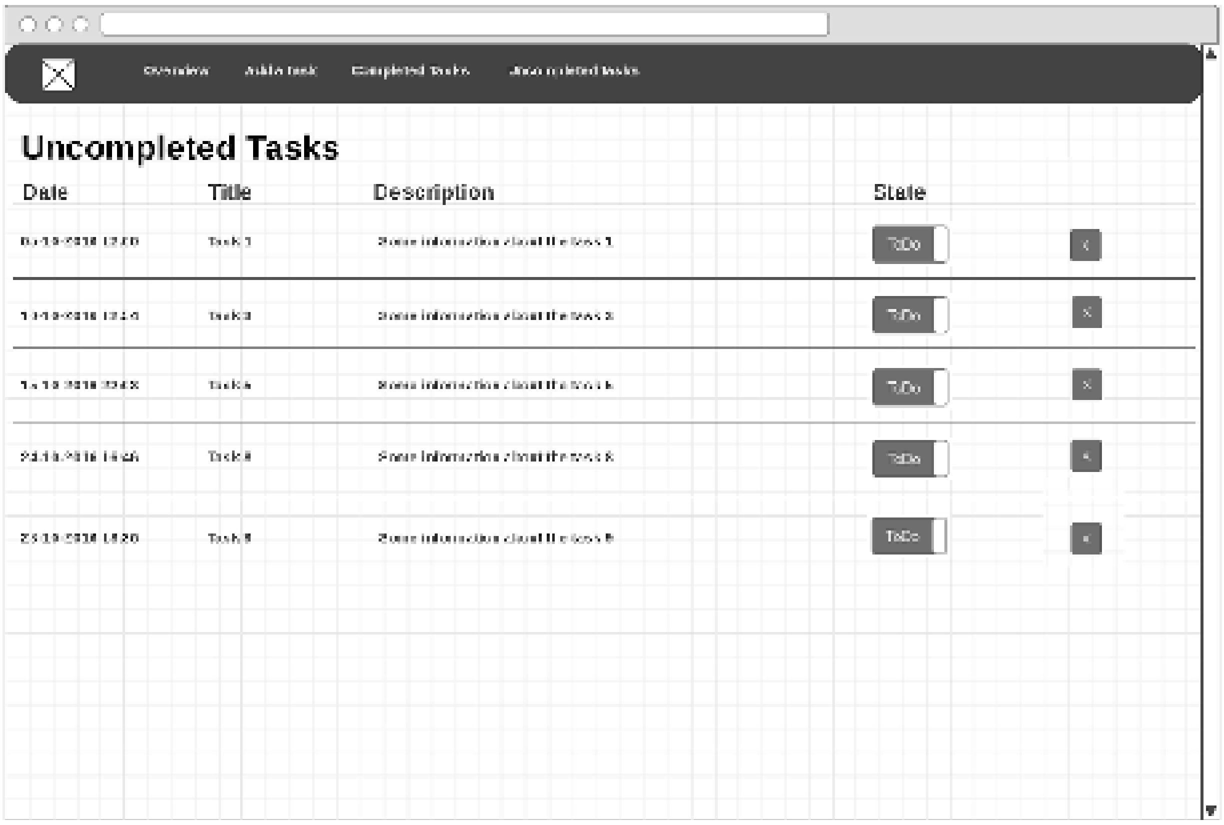
Overview: main page



Completed Tasks

Date	Title	Description	State	
[illegible]-10-2016 15:10	Task 2	Some information about the task 2	Done	X
[illegible]-10-2016 05:20	Task 4	Some information about the task 4	Done	X
[illegible]-10-2016 07:55	Task 6	Some information about the task 6	Done	X
[illegible]-10-2016 13:04	Task 7	Some information about the task 7	Done	X

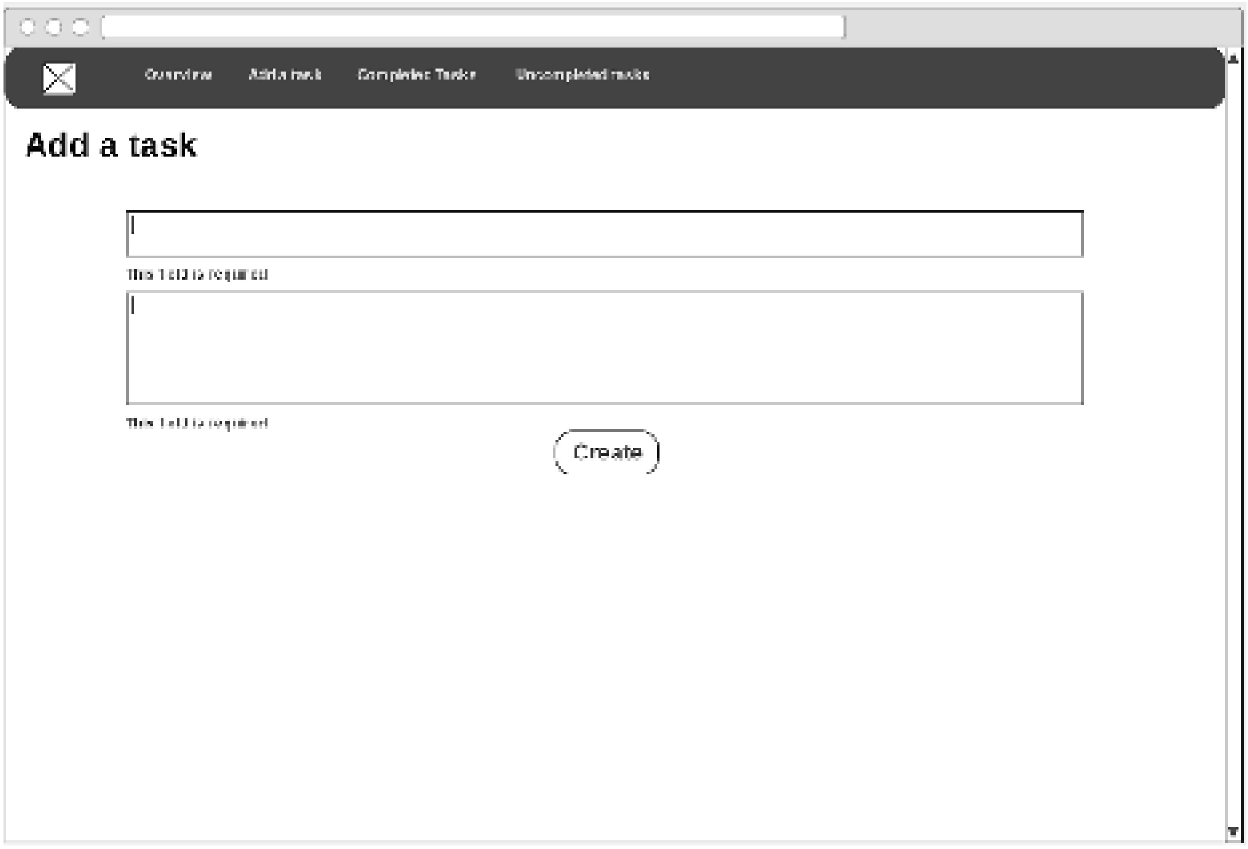
Completed Tasks: completed_tasks.html



Uncompleted Tasks

Date	Title	Description	State	
05-10-2016 12:20	Task 1	Some information about the task 1	ToDo	x
11-10-2016 12:40	Task 3	Some information about the task 3	ToDo	x
15-10-2016 22:45	Task 5	Some information about the task 5	ToDo	x
23-10-2016 16:45	Task 8	Some information about the task 8	ToDo	x
25-10-2016 18:20	Task 9	Some information about the task 9	ToDo	x

Uncompleted Task: uncompleted_tasks.html



Add a task

This field is required

This field is required

Add a Task: create_task.html

II. Choice of Layout

I chose to create a navigation bar to allow the user to navigate between the Overview, the Completed tasks, uncompleted tasks and Add a task pages. The navbar is fixed on scroll so if the user scrolls down the navigation bar will still be on top on the page and accessible. Thereby the user can always navigate between the different sections easily. I also added a header at just after the navbar so the user can always easily know where he is.

I chose to display the different task in an array with the following columns:

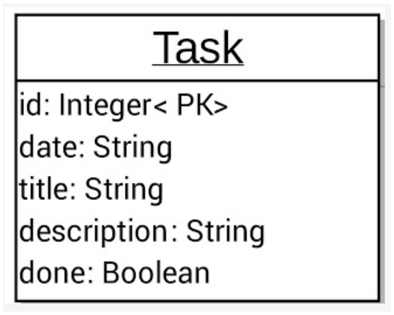
- **Date**: The date is composed of the day, month, year, hour and minutes like the following example: 13-10-2016 12:07
- **Title**
- **Description**
- **State**: This column contain a switch that allows the user to set the state of the tasks. The user can choose between "Done" and "ToDo"
- The last column doesn't have a header. It contains buttons that allow the user to delete a task

This table is used in the main page called Overview, in the Completed Tasks page and in the Uncompleted Tasks page.

The page Add a task is composed of two text input fields, one for the title of the task and one for the description. Then, the user has to press the button to create the task. If the fields are empty a message in red is displayed below the concerned text field.

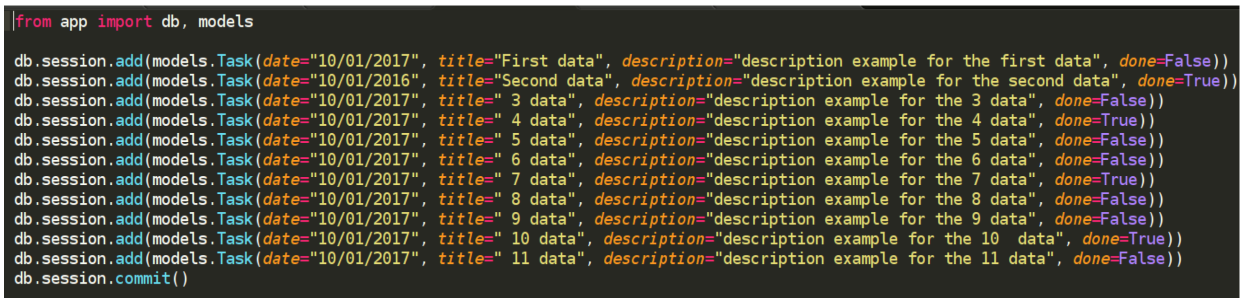
III. Database Model: Task

The database model implemented is called Task. This model contains a primary key id used to retrieve an element. Since the id is an integer value the type used is Integer. It also contains date used to store the creation date of the task as a String, a title and a description used to name the task and explain it and a boolean variable called done used to store the state of the task. If done is true the task is already done. If done is false the task isn't done yet.



IV. Evaluation

To create the ToDo List website, I first needed to create a database that can contain the different tasks. When the database was created, I had to test it. I first entered one task to check if the task was created and displayed in my html page. Then I created more tasks.

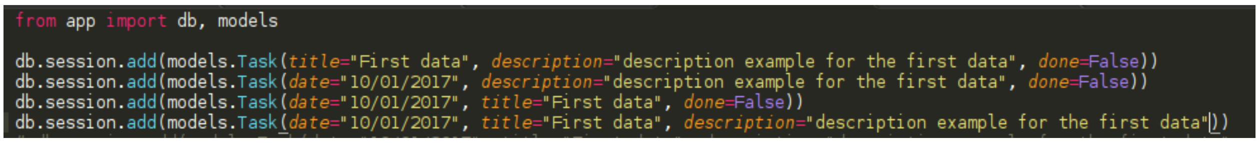


```
from app import db, models

db.session.add(models.Task(date="10/01/2017", title="First data", description="description example for the first data", done=False))
db.session.add(models.Task(date="10/01/2016", title="Second data", description="description example for the second data", done=True))
db.session.add(models.Task(date="10/01/2017", title=" 3 data", description="description example for the 3 data", done=False))
db.session.add(models.Task(date="10/01/2017", title=" 4 data", description="description example for the 4 data", done=True))
db.session.add(models.Task(date="10/01/2017", title=" 5 data", description="description example for the 5 data", done=False))
db.session.add(models.Task(date="10/01/2017", title=" 6 data", description="description example for the 6 data", done=False))
db.session.add(models.Task(date="10/01/2017", title=" 7 data", description="description example for the 7 data", done=True))
db.session.add(models.Task(date="10/01/2017", title=" 8 data", description="description example for the 8 data", done=False))
db.session.add(models.Task(date="10/01/2017", title=" 9 data", description="description example for the 9 data", done=False))
db.session.add(models.Task(date="10/01/2017", title=" 10 data", description="description example for the 10  data", done=True))
db.session.add(models.Task(date="10/01/2017", title=" 11 data", description="description example for the 11 data", done=False))
db.session.commit()
```

Python script used to a few tasks

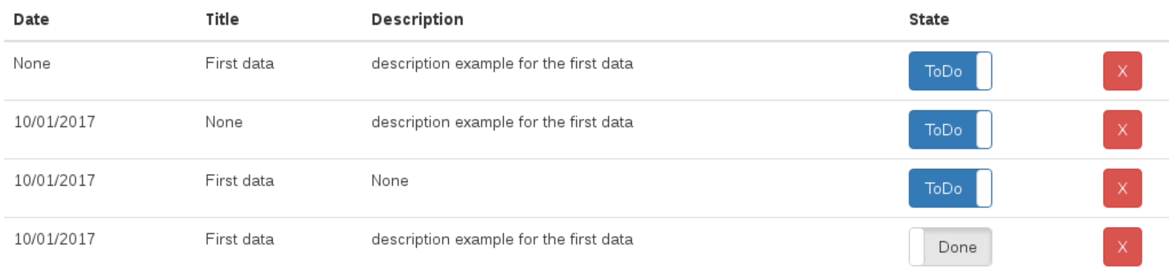
The database was working with “classic” values so I began to test it with wrong values. I first created tasks without a date, then without a title, without a description and I finished by creating tasks without a state (done or not). In all the cases, the missing value was set to None. Since the date doesn’t have to be provided by the user, I changed my python script so that the data is retrieved by the website and directly formatted and entered into the database.
When the state wasn’t provided, the task displayed with the state “true” (done). So I decided to automatically set the state to false when the task is created since the user hasn’t done it yet.



```
from app import db, models

db.session.add(models.Task(title="First data", description="description example for the first data", done=False))
db.session.add(models.Task(date="10/01/2017", description="description example for the first data", done=False))
db.session.add(models.Task(date="10/01/2017", title="First data", done=False))
db.session.add(models.Task(date="10/01/2017", title="First data", description="description example for the first data"))
```

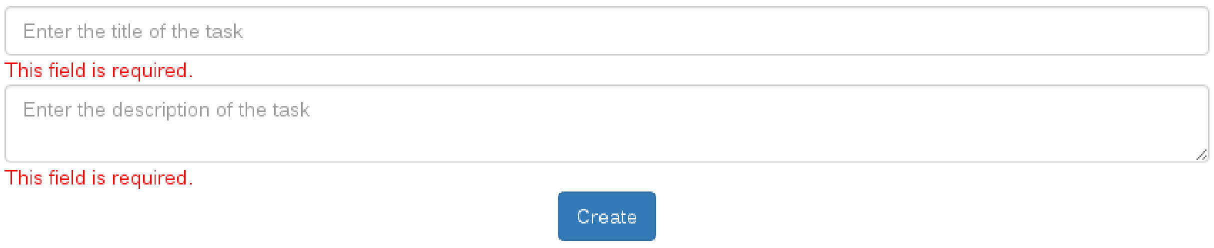
Python script used to add wrong tasks



Date	Title	Description	State	
None	First data	description example for the first data	ToDo	X
10/01/2017	None	description example for the first data	ToDo	X
10/01/2017	First data	None	ToDo	X
10/01/2017	First data	description example for the first data	Done	X

Result of the test

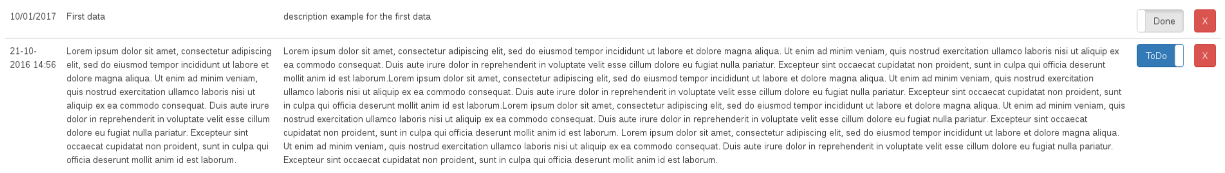
I also added validators so that the user has to enter a title and a description. At this stage of the development, the task is necessarily created with all the values.



Enter the title of the task
This field is required.
Enter the description of the task
This field is required.
Create

Error displayed to the user when fields are missing

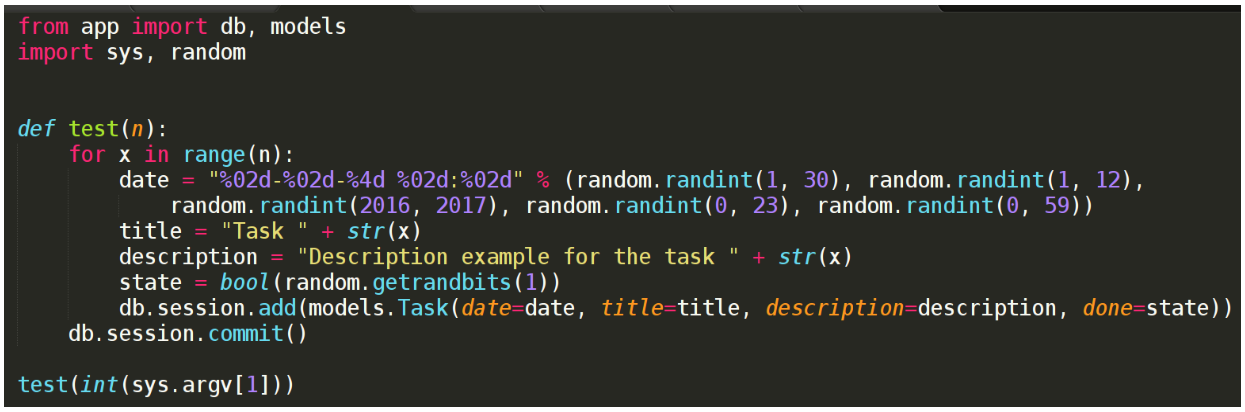
Then I needed to test the task creation with wrong datatypes or with title and description too long. In both cases the task was created with no error but the problem was to display it. When the title or the description where too long, the table was stretched and the user had to scroll all the way to be able to access the buttons. I fixed this problem by using the class “table-responsive”.



10/01/2017	First data	description example for the first data	Done	X
21-10-2016 14:56	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.	ToDo	X

Long tasks can be displayed fully without scrolling thanks to the table-responsive class

To test the database I also wrote a python script that generates tasks and adds them to the database. I try with 1000 tasks and the website was still working and displaying the tasks but it was taking more time to load.

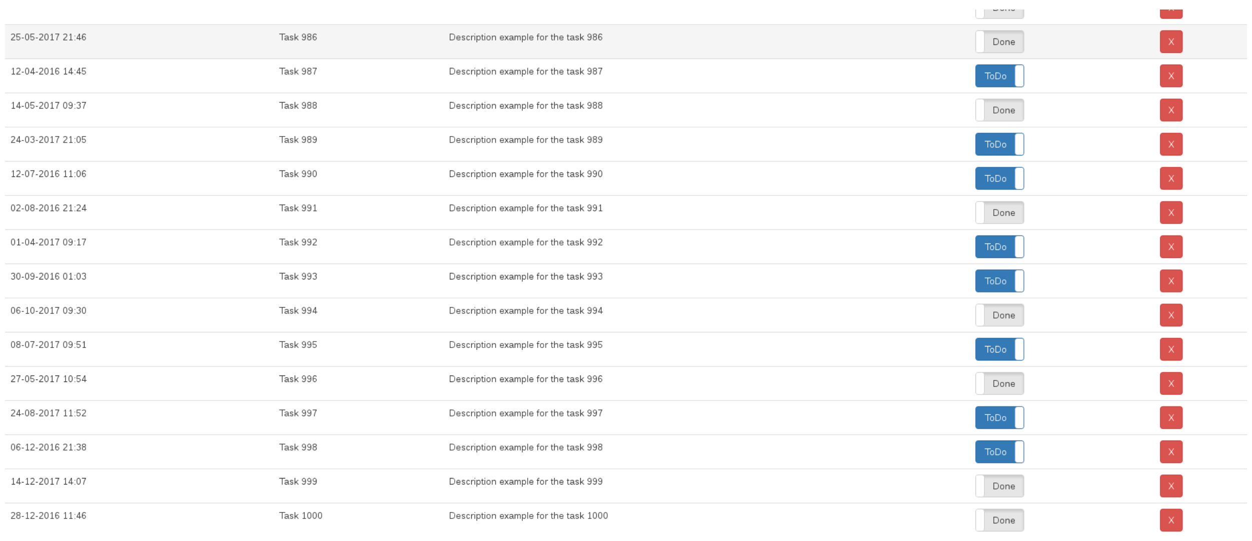


```python
from app import db, models
import sys, random


def test(n):
    for x in range(n):
        date = "%02d-%02d-%4d %02d:%02d" % (random.randint(1, 30), random.randint(1, 12),
            random.randint(2016, 2017), random.randint(0, 23), random.randint(0, 59))
        title = "Task " + str(x)
        description = "Description example for the task " + str(x)
        state = bool(random.getrandbits(1))
        db.session.add(models.Task(date=date, title=title, description=description, done=state))
    db.session.commit()

test(int(sys.argv[1]))
```

Python SCript used to generate a given number of tasks



25-05-2017 21:46	Task 986	Description example for the task 986	Done	X
12-04-2016 14:45	Task 987	Description example for the task 987	ToDo	X
14-05-2017 09:37	Task 988	Description example for the task 988	Done	X
24-03-2017 21:05	Task 989	Description example for the task 989	ToDo	X
12-07-2016 11:06	Task 990	Description example for the task 990	ToDo	X
02-08-2016 21:24	Task 991	Description example for the task 991	Done	X
01-04-2017 09:17	Task 992	Description example for the task 992	ToDo	X
30-09-2016 01:03	Task 993	Description example for the task 993	ToDo	X
06-10-2017 09:30	Task 994	Description example for the task 994	Done	X
08-07-2017 09:51	Task 995	Description example for the task 995	ToDo	X
27-05-2017 10:54	Task 996	Description example for the task 996	Done	X
24-08-2017 11:52	Task 997	Description example for the task 997	ToDo	X
06-12-2016 21:38	Task 998	Description example for the task 998	ToDo	X
14-12-2017 14:07	Task 999	Description example for the task 999	Done	X
28-12-2016 11:46	Task 1000	Description example for the task 1000	Done	X

1000 tasks created