

**Candidato:** Mattia Ruberto  
**Azienda:** SAMT  
**Data inizio:** 03.09.2019  
**Data di consegna:** 20.12.2019

### **Situazione iniziale**

Lo scopo di questo progetto è quello di poter gestire un negozio di acquari marini tramite un applicazione web. Del negozio bisogna gestire i dipendenti che devono avere accesso al sito, le vasche che possono essere modificate da tutti ma aggiunte e rimosse solo dall'amministratore. Stessa cosa per la gestione degli utenti a cui può avere accesso solamente l'amministratore. Poi bisogna gestire gli abitanti della vasca, a questa pagina ci possono accedere tutti. Realizzando questo prodotto sarebbe possibile passare da un'eventuale gestione su carta o su un file excel a una gestione molto più facile da gestire, perché tutte le attività di aggiunta, modifica e rimozione sono facilitate, molto più facile da reperire essendo che ovunque tu sei è possibile accedere al sito ed effettuare tutte le operazioni necessarie.

### **Attuazione**

Per realizzare il seguente progetto ho usato la struttura MVC, questo per il semplice fatto che mi trovo bene nell'utilizzarla e ormai ci ho fatto l'abitudine dato che semplifica e organizza la gestione di tutte le classi in blocchi ben separati e riconoscibili che ti permettono di avere sempre un certo ordine sia nel progetto che nella tua testa e quindi di non avere problemi che potrebbero essere dovuti alla cattiva organizzazione. Per l'implementazione del frontend ho usato un template che implementa Bootstrap, anche qua per un fattore di comodità essendo che tutto ciò che devo fare lo posso fare tranquillamente con l'utilizzo di questa libreria molto semplice da utilizzare allo stesso tempo ti permette di fare tante cose.

### **Risultati**

I requisiti che sono stati richiesti all'inizio del progetto sono stati raggiunti, il sito web funziona ed è pronto all'uso senza nessun tipo di problema. Il proprietario di un negozio di acquari marini potrà quindi gestire dipendenti, acquari e abitanti tutti in un unico posto in modo veloce e facile.

## Gestione Acquari Marini

**Titolo del progetto:** Gestione Acquari Marini  
**Alunno/a:** Mattia Ruberto  
**Classe:** I4AA  
**Anno scolastico:** 2019/2020  
**Docente responsabile:** Luca Peduzzi

1	Introduzione .....	3
1.1	Informazioni sul progetto .....	3
1.2	Abstract .....	3
1.3	Scopo .....	3
2	Analisi.....	4
2.1	Analisi del dominio .....	4
2.2	Analisi e specifica dei requisiti.....	4
2.3	Use case .....	7
2.4	Pianificazione .....	8
2.5	Analisi dei mezzi.....	1
2.5.1	Software .....	1
2.5.2	Hardware.....	1
3	Progettazione .....	1
3.1	Design dell'architettura del sistema.....	1
3.2	Design dei dati e database.....	2
3.3	Design delle interfacce .....	3
3.3.1	Pagina di login.....	3
3.3.2	Pagina nuova password .....	3
3.3.3	Pagina riassuntiva .....	4
3.3.4	Pagina gestione utenti .....	5
3.3.5	Pagina gestione vasche .....	5
3.3.6	Pagina gestione vasca .....	6
4	Implementazione .....	7
4.1	Design dell'architettura.....	7
4.2	Database .....	7
4.3	Frontend.....	8
4.3.1	Pagina login.....	8
4.3.2	Pagina nuova password .....	8
4.3.3	Pagina riassuntiva .....	9
4.3.4	Pagina gestione vasche .....	9
4.3.5	Pagina gestione utenti .....	10
4.3.6	Pagina gestione abitanti .....	10
4.3.7	Form gestione vasche .....	11
4.3.8	Form gestione utenti.....	11
4.4	Backend .....	12
4.4.1	Model.....	13
4.4.1.1	MailModel .....	13
4.4.1.2	Database .....	14
4.4.1.3	ValidationFunction.....	15
4.4.1.4	HabitantValidation .....	17
4.4.1.5	UserModel .....	18
4.4.2	Controller.....	21
4.4.2.1	Login.....	21
4.4.2.2	Home.....	22
4.4.2.3	NewPassword .....	23
4.4.2.4	TankManagement .....	24
4.4.2.5	UserManagement .....	27
4.4.2.6	HabitantManagement .....	29
4.4.2.7	TankValueControl.....	32
4.5	Protocollo di test.....	34
4.6	Risultati test.....	37
5	Consuntivo .....	1
6	Conclusioni.....	1
6.1	Sviluppi futuri.....	1
6.2	Considerazioni personali .....	1
7	Bibliografia .....	2
7.1	Sitografia .....	2
8	Allegati .....	2

## **1 Introduzione**

---

### **1.1 Informazioni sul progetto**

Allievo: Mattia Ruberto  
Docente: Luca Peduzzi  
Classe: I4AA  
Anno scolastico: 2019-2020  
Scuola: Arti e mestieri Trevano  
Sezione Informatica  
Data inizio: 03.09.2019  
Data fine: 20.12.2019

### **1.2 Abstract**

This documentation contains the entire process of planning and implementing the development of a web application system that allows the staff of a marine aquarium shop to manage the maintenance of the tanks keeping the data under control by updating the website that has a summary page of all the aquaria and that directly manages and alerts the administrator if the water values of a particular tank are incorrect or if weekly changes have not been made.

### **1.3 Scopo**

Lo scopo del progetto è di realizzare un applicativo web che gestisce la manutenzione degli acquari marini di un negozio, questo permette agli utenti che saranno il personale del negozio di effettuare tutti i test dell'acqua e i cambi settimanali di tutte le vasche presenti consentendo di avere una pagina semplice e pratica dove ci sono elencate tutte le informazioni degli acquari e di segnalare tutte le attività di manutenzione che non sono state fatte per evitare dimenticanze da parte del personale oltre tutto anche l'amministratore o il capo del negozio può tenere tutto sotto controllo tramite sito web avendo così a disposizione una pagina riassuntiva sempre online e aggiornata sullo stato del negozio.

## 2 Analisi

### 2.1 Analisi del dominio

Il seguente progetto deve gestire tramite un applicativo via web e email le vasche di un negozio di acquari marini, di esso deve essere gestita la manutenzione settimanale degli acquari tramite degli appositi test per l'acqua, i parametri essenziali da essere misurati per il corretto funzionamento del sistema sono il magnesio, il calcio e il Kh. Oltre ai test dell'acqua bisogna gestire anche i cambi d'acqua che vengono effettuati ogni 14 giorni e gli abitanti delle vasche. Questo permette al personale del negozio di poter tenere sotto controllo tutte le vasche di esso evitando qualche dimenticanza che potrebbe nuocere agli esseri viventi presenti nelle vasche. Nel sito ci sono gli utenti normali che possono aggiungere solo i valori dell'acqua e i cambi che sono stati effettuati e un amministratore che deve gestire le vasche presenti nel negozio e gli utenti. Gli utenti che andranno ad utilizzare questo prodotto non saranno degli esperti del settore quindi il prodotto ovviamente deve essere user-friendly, per operare nel prodotto non bisogna avere nessuna competenza specifica ma bisogna saper fare l'accesso sul sito web e disporre di account di posta elettronica.

### 2.2 Analisi e specifica dei requisiti

ID: REQ-01	
<b>Nome</b>	Sito web
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Il sito web deve essere responsive.
<b>002</b>	User-friendly, il sito web deve essere facile da utilizzare.
<b>003</b>	Attendibilità dei dati inseriti dall'utente.

ID: REQ-02	
<b>Nome</b>	Pagina login
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Ci devono essere due campi di richiesta: <ul style="list-style-type: none"> <li>• Username che sarà l'email</li> <li>• Password</li> </ul>
<b>002</b>	Se viene inserita la password fornita dal sistema deve permettere di farla cambiare all'utente creando un nuovo campo dove immettere la nuova password.
<b>003</b>	In caso di perdita di password l'utente deve poter chiedere una nuova password.

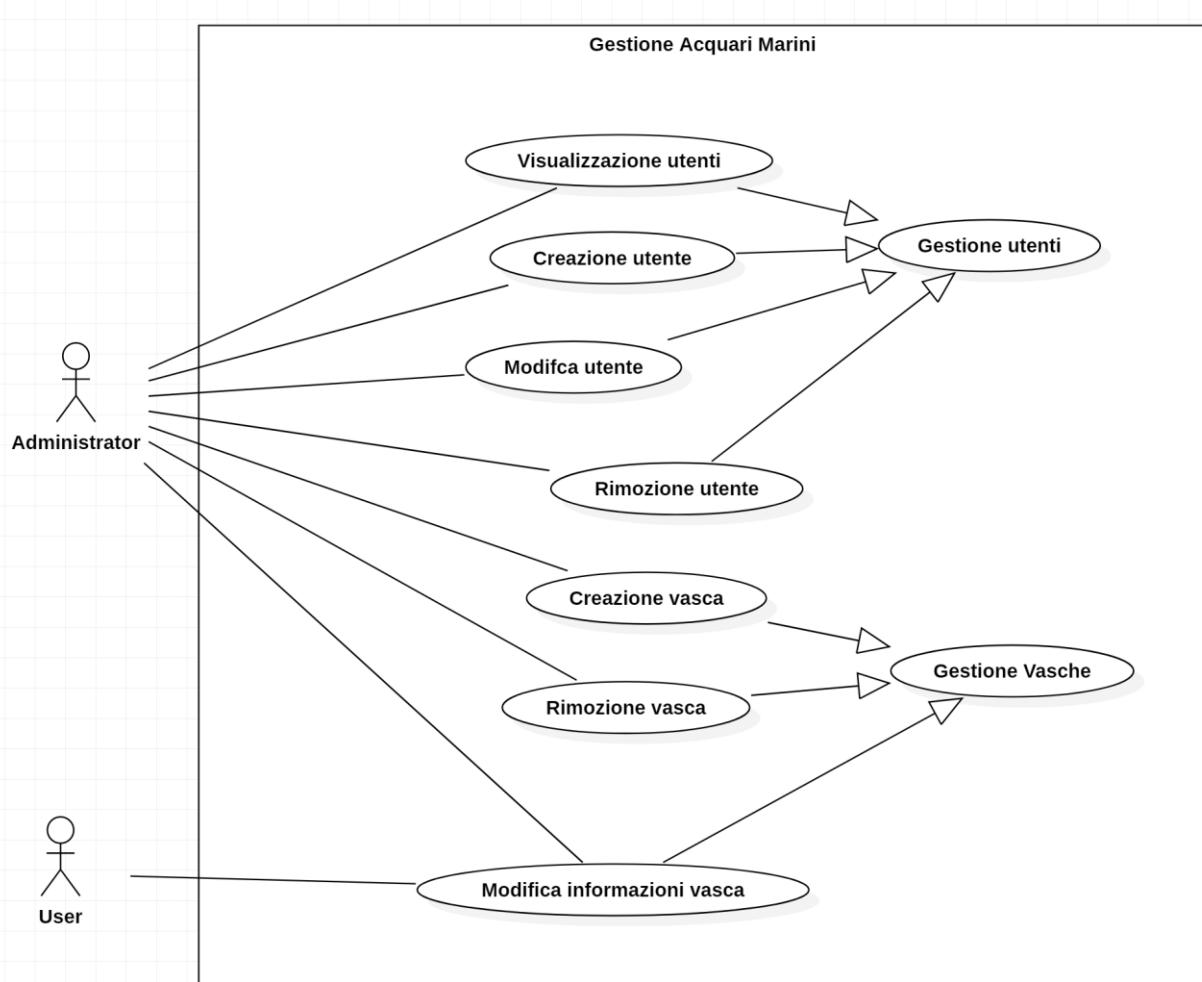
ID: REQ-03	
<b>Nome</b>	Pagina amministrazione vasche
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	L'amministratore deve poter inserire, modificare e cancellare le vasche presenti nel negozio, l'utente solamente modificare.
<b>002</b>	Per ogni vasca deve specificare i parametri: <ul style="list-style-type: none"> <li>• I litri totali della vasca</li> <li>• Il livello di magnesio</li> <li>• Il livello di calcio</li> <li>• Il livello di Kh</li> </ul>
<b>003</b>	Per ogni vasca deve specificare gli abitanti: <ul style="list-style-type: none"> <li>• Nome</li> <li>• Se è un pesce, un crostaceo o un Corallo</li> <li>• Se è maschio o femmina</li> </ul>

ID: REQ-04	
<b>Nome</b>	Pagina amministrazione utenti
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	L'amministratore deve poter creare, cancellare e modificare gli utenti e i loro permessi.
<b>002</b>	Quando l'amministratore crea un nuovo utente deve mettere un indirizzo email e la password che verrà assegnata in automatico da un generatore random.
<b>003</b>	Un amministratore può far diventare in qualsiasi momento un utente amministratore o un utente con diritti limitati cioè solo di aggiornamento valori.
<b>004</b>	Quando l'amministratore crea un nuovo utente o aggiorna la password essa dovrà essergli spedita direttamente per email.

ID: REQ-05	
<b>Nome</b>	Pagina riassunto giornaliero
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Deve tenere presente i valori aggiornati: <ul style="list-style-type: none"> <li>• Valori dell'acqua</li> <li>• Abitanti della vasca</li> </ul>
<b>002</b>	Deve tenere della soglia minima e massima dei parametri ed inviare un email all'amministratore se i valori raggiungono un livello critico: <ul style="list-style-type: none"> <li>• Magnesio: min. 1200 – max. 1450</li> <li>• Calcio: min. 350 – max. 450</li> <li>• Kh: min. 7 – max. 11</li> </ul>
<b>003</b>	Deve tenere in conto la diminuzione giornaliera dei valori dell'acqua: <ul style="list-style-type: none"> <li>• Magnesio: 100</li> <li>• Calcio: 25</li> <li>• Kh: 1</li> </ul>

ID: REQ-06	
<b>Nome</b>	Database
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
<b>001</b>	I dati devono essere salvati su un database.
<b>002</b>	Non deve essere possibile effettuare attacchi sul database

## 2.3 Use case



I casi d'uso di questa applicazione sono principalmente divisi in due parti, il primo riguarda se l'utente che accede ha gli accessi come amministratore infatti esso può gestire tutti gli utenti presenti e quindi può crearli, eliminarli e modificare le loro informazioni compresi i permessi dato che un amministratore può far diventare un utente normale amministratore e un amministratore un utente normale. Inoltre può creare nuove vasche all'interno del sito da gestire o rimuoverle e di conseguenza può anche gestire i valori di ogni vasca. Mentre gli utenti normali possono leggere solamente le informazioni riguardanti le varie vasche ma non possono aggiungerle o rimuoverle e di conseguenza non possono neanche gestire gli utenti che non vedono neanche.

## 2.4 Pianificazione

Le principali categorie che compongono questo gantt sono:

- Documentazione → comprende tutto ciò che metterò nella documentazione.
- Analisi → comprende tutte quelle attività che permettono di capire i requisiti e ciò che viene richiesto dal committente.
- Progettazione → tutta la parte di progettazione come design delle interfacce, design del database, use case, ...
- Implementazione → tutta la parte implementativa del progetto come scrittura codice, creazione pagine, creazione database, ...
- Test → Parte che comprende i test che permettono di capire se l'applicazione rispetta tutti i requisiti e fa quello che deve fare.
- Consegna progetto → fine progetto, giorno di consegna.

Nell'analisi fanno parte le seguenti categorie:

- Lettura dei requisiti → lettura del quaderno dei compiti consegnatoci dal committente in questo caso il formatore.
- Colloquio con il committente → colloquio con il committente dove fai tutte le domande per risolvere tutti i dubbi che sono venuti fuori leggendo il quaderno dei compiti.
- Stesura gantt → pianificazione del progetto e tutte le varie tappe.
- Analisi di dominio → analisi dell'applicazione, di cosa deve fare e per chi deve farlo.
- Analisi dei requisiti → individuare tutti i requisiti che ci sono nel quaderno dei compiti.

Nella progettazione ci sono i seguenti compiti:

- Design dell'architettura → piccolo schema che riguarda la struttura dell'applicazione.
- Design del database → diagramma ER del data base.
- Design delle interfacce → design di tutte le interfacce in questo caso pagine del sito.
- Use case → diagramma uml per sintetizzare tutte le operazioni che vengono fatte dai vari utenti.

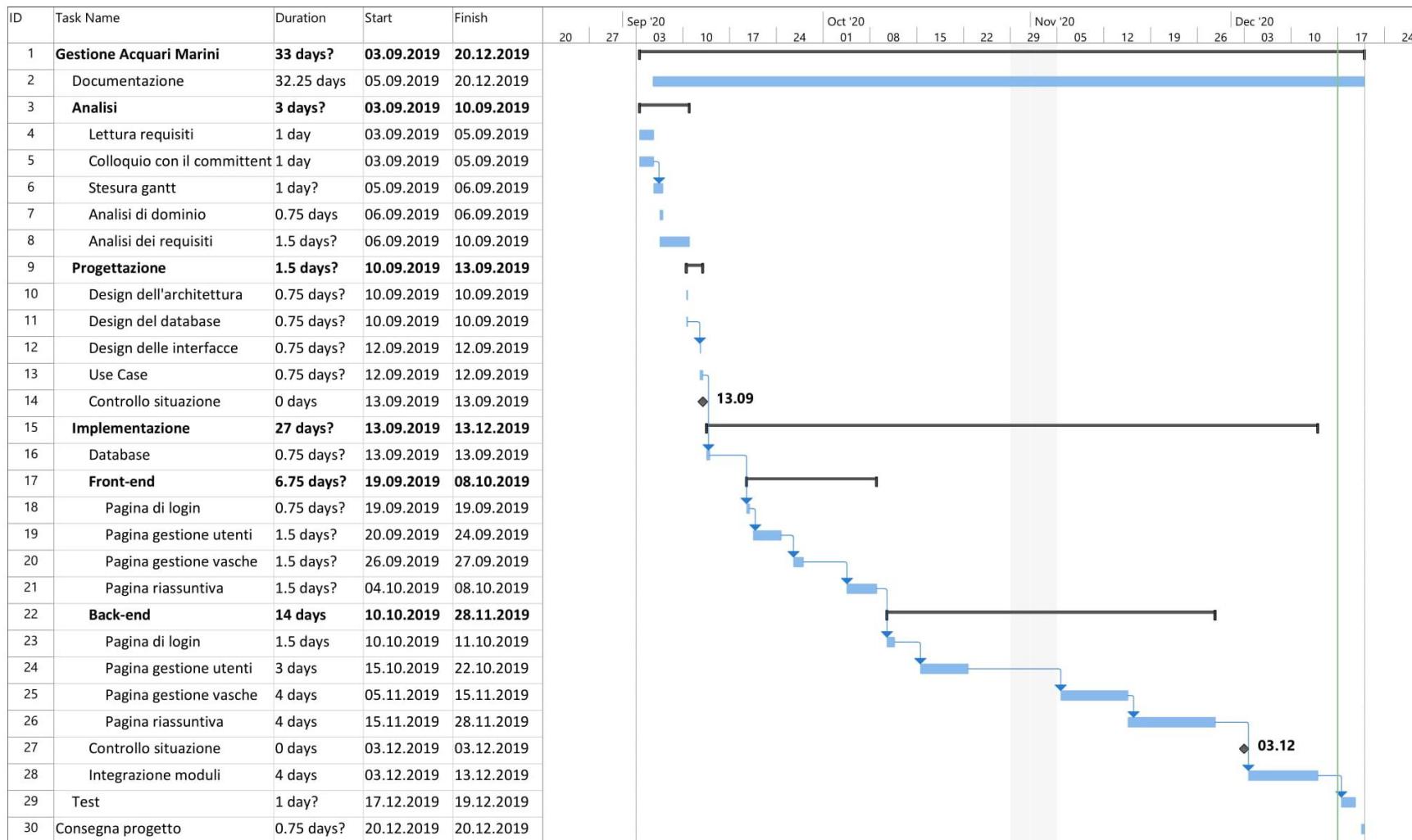
Nell'implementazione ci sono tre categorie principali:

- Frontend → implementazione di tutta la parte grafica dei vari moduli che compongono il prodotto.
- Backend → implementazione di tutta la parte logica dei vari moduli che compongono il prodotto.
- Integrazione moduli → fase in cui si prendono tutti i moduli e si integrano per andare a formare l'applicazione.

Nella fase di test viene testata l'applicazione per controllare che tutti i requisiti siano rispettati e l'applicazione faccia quello che è stato richiesto dal committente.

Ho messo due milestone, la prima l'ho piazzata prima di incominciare a implementare e la seconda prima di integrare tutti i moduli per essere sicuro di aver implementato tutto il frontend e il backend.

Questa è praticamente la prima versione di gantt infatti ho tolto solo una milestone dato, ho tolto l'attività presentazione dato che siamo stati informati dopo che non era da inserire e ho cambiato il nome e la data di un attività di 4 ore.



**Titolo del progetto:** Gestione Acquari Marini  
**Alunno/a:** Mattia Ruberto  
**Classe:** I4AA  
**Anno scolastico:** 2019/2020  
**Docente responsabile:** Luca Peduzzi

## 2.5 Analisi dei mezzi

Per la realizzazione di questo progetto non sono serviti mezzi particolari ma un semplicissimo computer e i software di base.

### 2.5.1 Software

- Project Professional 2016
- Sublime Text 3
- Word 2016
- PowerPoint 2016
- Notepad++
- StarUML

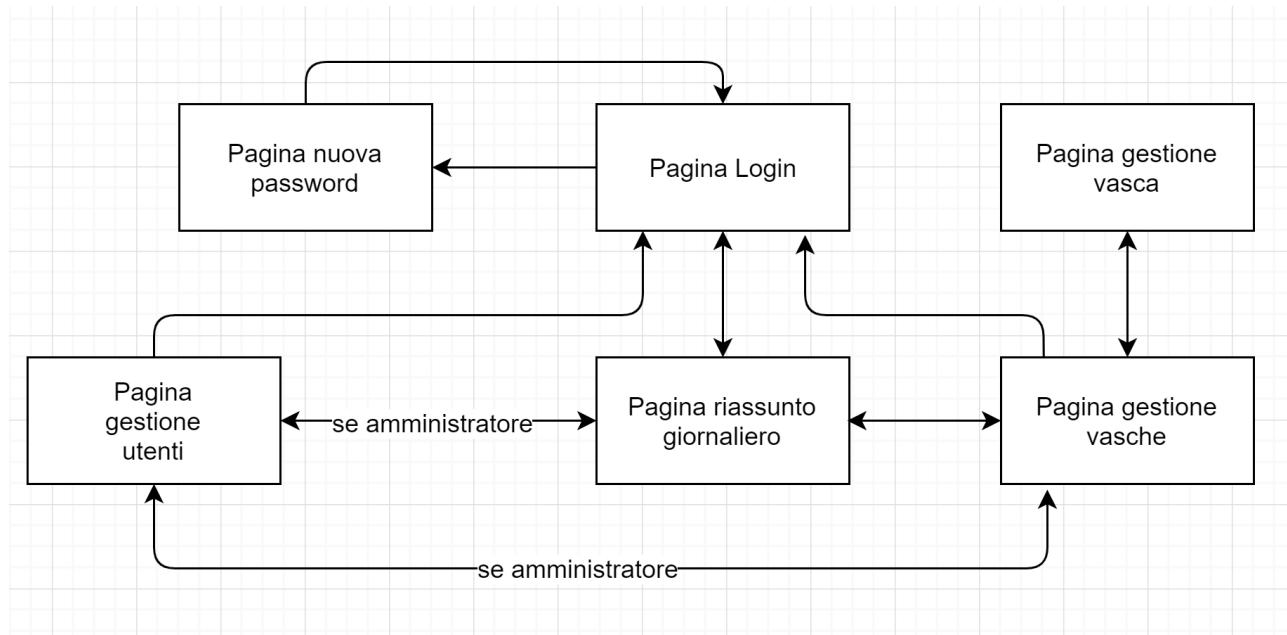
### 2.5.2 Hardware

- Computer personale:
  - Processore: Intel® Core™ i7-6700HQ CPU 2.60GHz
  - RAM: 16.0 GB DDR4
  - Sistema: 64bit, processore x64
  - Sistema operativo: Windows 10

## 3 Progettazione

---

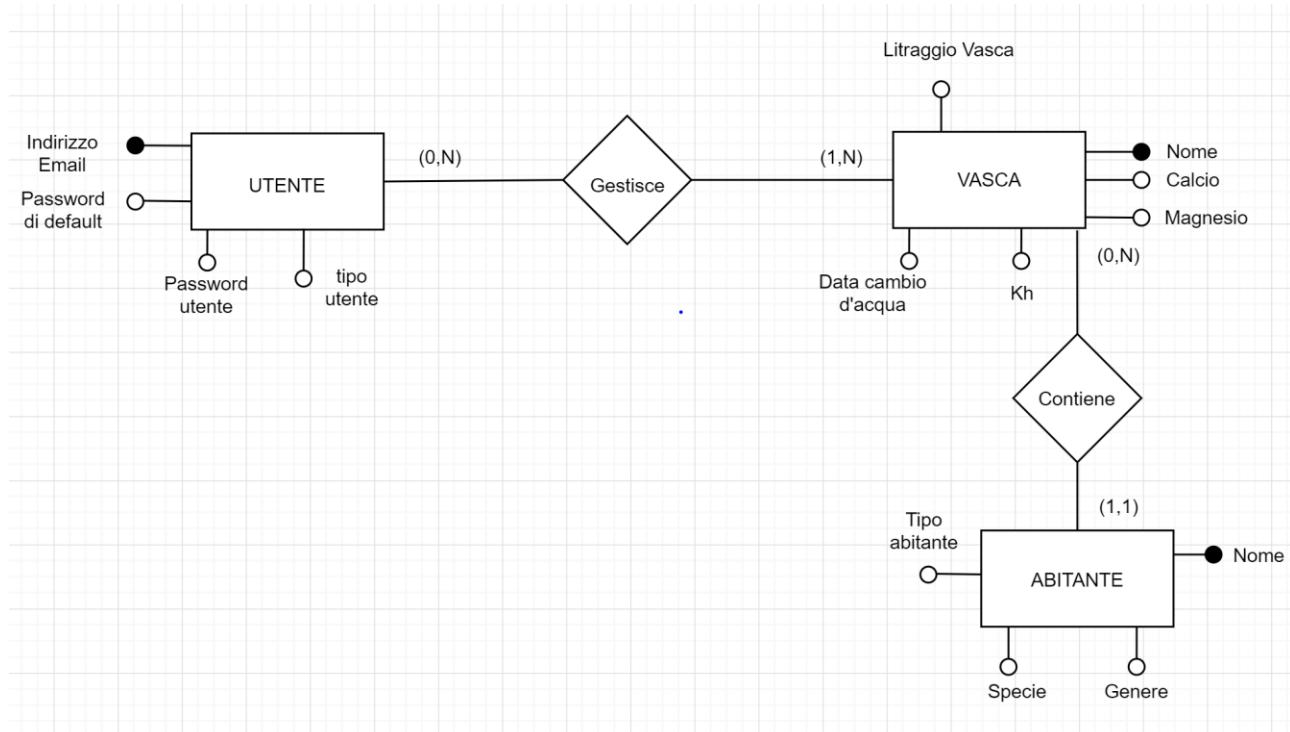
### 3.1 Design dell'architettura del sistema



**Titolo del progetto:** Gestione Acquari Marini  
**Alunno/a:** Mattia Ruberto  
**Classe:** I4AA  
**Anno scolastico:** 2019/2020  
**Docente responsabile:** Luca Peduzzi

Nella struttura del sito web ci sarà una pagina di login dove l'utente potrà inserire le sue credenziali, se l'utente inserisce la password di default il sito lo porterà alla pagina dedicata a poter inserire la nuova password, dopodiché verrà riportato di nuovo al login dove potrà accedere con la password nuova. Una volta effettuato l'accesso la prima pagina che comparirà sarà la pagina riassuntiva dove ci sarà una tabella contenente tutte le vasche con i loro. Da questa pagina sarà possibile tramite il menu spostarsi nella pagina gestione utenti se si è admin o nella pagina gestione vasche, e da qui nella pagina gestione vasca.

### 3.2 Design dei dati e database



Nel seguente database che verrà utilizzato per questo prodotto ci sarà una tabella utente che memorizzerà le informazioni dell'utente, in particolare l'email, la password di default e la password dell'utente che però non sarà possibile visualizzare. In un'altra tabella verrà memorizzato il tipo di utente che è collegata direttamente a esso quindi l'utente potrà avere un massimo di un tipo utente cioè i permessi. Ogni utente può gestire da 0 a più vasche e per ognuna di essa viene memorizzato il nome, il litraggio, la data dell'ultimo cambio d'acqua, il livello di calcio, di magnesio e kh. Ogni vasca può contenere da 0 a più abitanti, per ognuno di esso viene memorizzato il nome e il sesso. Ogni abitante ha un tipo di abitante cioè pesce, corallo o crostaceo e può essere solamente uno di questi ovviamente.

### 3.3 Design delle interfacce

In questo capitolo vengono rappresentate come sono state progettate prima dell'implementazione le pagine del sito web in linea generale infatti queste possono essere cambiate durante l'implementazione.

#### 3.3.1 Pagina di login

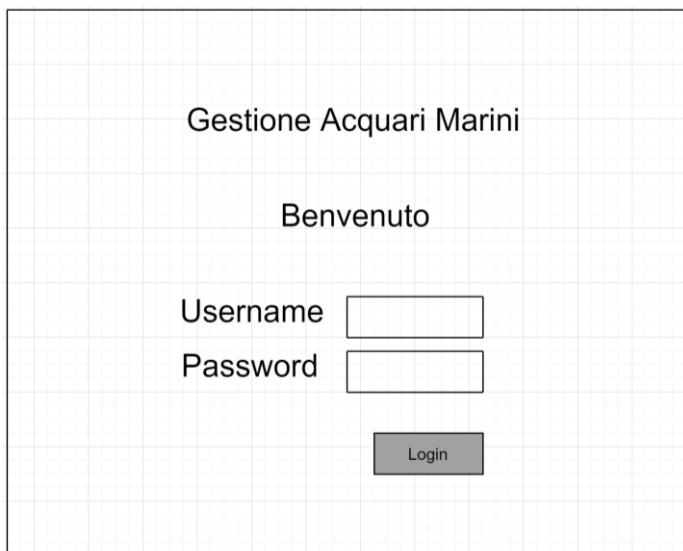


Diagramma della pagina di login:

La pagina ha un titolo "Gestione Acquari Marini" e un messaggio di benvenuto "Benvenuto". Contiene campi per l'username e la password, e un bottone "Login".

Username	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Login"/>	

Questa sarà la pagina di login dove l'utente potrà inserire le proprie credenziali nei due campi username e password e dopo aver schiacciato il bottone login il sito confronterà le credenziali immesse con quelle del database e se corrispondono porterà l'utente alla pagina di registrazione mentre se la password è di default porta l'utente in un'altra pagina dove può inserire le nuove credenziali.

#### 3.3.2 Pagina nuova password

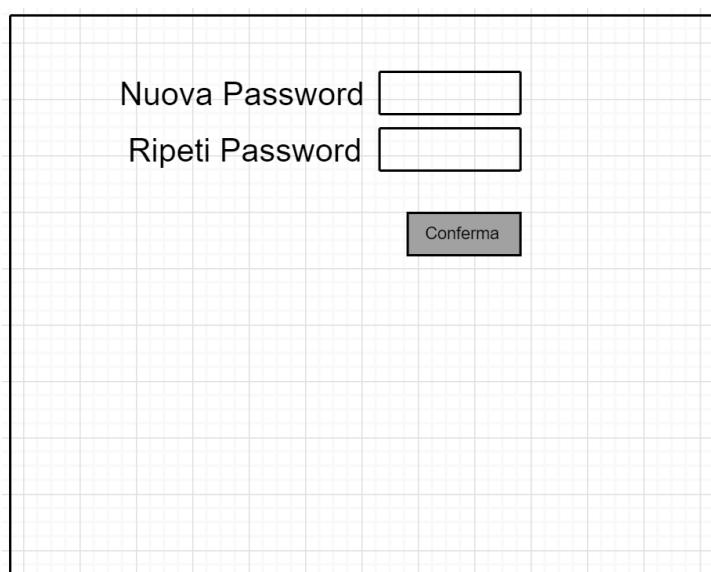


Diagramma della pagina di nuova password:

La pagina contiene campi per la nuova password e la ripetuta password, e un bottone "Conferma".

Nuova Password	<input type="text"/>
Ripeti Password	<input type="text"/>
<input type="button" value="Conferma"/>	

Questa pagina è molto semplice infatti permette all'utente semplicemente di cambiare la password di default con una personale. Quando al login l'utente inserisce la password di default il sito ti porta direttamente a questa pagina dove devi inserire una nuova password, dopodiché si schiaccia il bottone conferma che ti riporta alla pagina di login ed è possibile rieffettuare il login con la nuova password.

### **3.3.3 Pagina riassuntiva**

La pagina riassuntiva è la prima pagina che compare dopo che l'utente ha eseguito il login se va tutto bene con le credenziali. Nella pagina riassuntiva ci sarà una tabella che rappresenta tutte le vasche con le loro informazioni sui valori dell'acqua e la data dell'ultimo cambio d'acqua inoltre avviserà anche l'utente in rosso i valori che stanno uscendo la range ideale per lo stato degli abitanti delle varie vasche. Da questa pagina sarà possibile muoversi nella pagina di amministrazione se si hanno i permessi giusti, nella pagina della gestione delle vasche o tornare al login.

### **3.3.4 Pagina gestione utenti**

Questa pagina permette all'amministratore di aggiungere, eliminare e modificare gli utenti che possono accedere al sito, infatti in questa pagina si potrà accedere solo con i permessi di amministratore e tutte le modifiche riguardanti gli utenti si possono fare da qui senza andare direttamente sul database. Ci sarà un tabella che rappresenta gli utenti con la loro email, password e i loro permessi. Da questa pagina si può tornare al login, alla pagina di gestione delle vasche e degli utenti.

### **3.3.5 Pagina gestione vasche**

In questa pagina ci potranno arrivare tutti gli utenti e conterrà una tabella con il nome della vasca dove si potrà scegliere la vasca scelta e cliccandoci sopra sarà possibile aprire una nuova pagina sulla gestione personale della singola vasca. In questa pagina solo gli utenti amministratori potranno aggiungere vasche o eliminarle. Da questa pagina si può tornare al login, alla pagina di gestione degli utenti.

### 3.3.6 Pagina gestione vasca

Pagina gestione vasche

## Pagina Gestione Vasca

Aggiungi abitante

Nome	Sesso	Tipo	Modifica	Elimina
			[Button]	[Button]
....	....	....	....	....

**Informazioni**

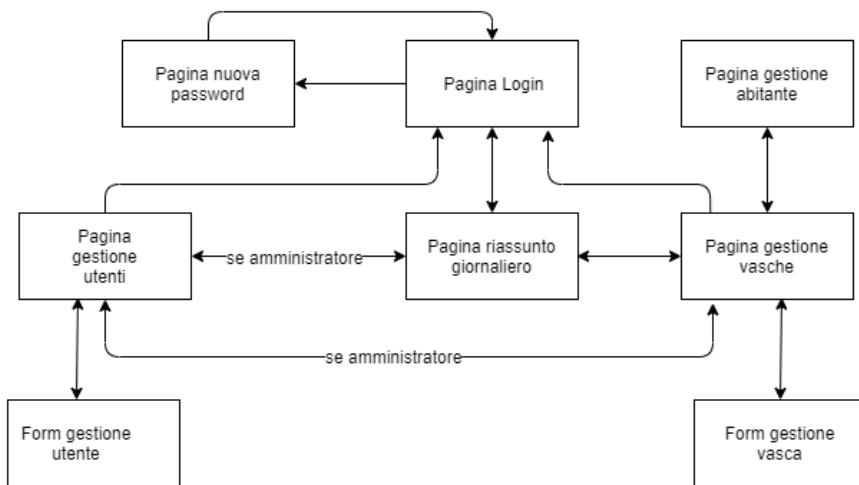
Magnesio	Calcio	Kh	Modifica
			[Button]
....	....	....	....

Questa pagina compare dopo che nella gestione delle vasche viene cliccata una vasca specifica, a questo punto il sito ti mostra in una tabella tutti gli abitanti della vasca che possono essere pesci, corstacei e coralli. È possibile eliminarli, aggiungerli o modificarli. E in un'altra tabella ci saranno i valori dell'acqua dell'acquario e l'ultimo cambio d'acqua effettuato che potrà essere solo modificato e non eliminato o aggiunto ovviamente. Da questa pagina si può tornare solo alla pagina della gestione di tutte le vasche.

## 4 Implementazione

### 4.1 Design dell'architettura

Per quanto riguarda il design del sito web è un po' cambiato rispetto la progettazione, infatti durante l'implementazione per un motivo di estetica o di praticità ho cambiato alcune cose



Come si può vedere ci sarà solamente una pagina gestione vasca, in questa pagina potrò aggiungere e modificare le vasche, sia per l'aggiunta che per la modifica verrà utilizzato un singolo form, dove ci saranno tutti i campi da riempire. Stessa cosa per la gestione degli utenti. Per quanto riguarda gli abitanti, nella pagina gestione vasche è presente un bottone per ogni vasca che porta a una pagina di gestione degli abitanti dove si possono aggiungere, modificare e eliminare, per gli abitanti il form di gestione è nella stessa pagina.

### 4.2 Database

Per l'implementazione del database che è in MySQL ho usato il terminale o MySQLWorkbench.

vasca		abitante		utente	
nome	varchar(45)	specie	varchar(45)	email	varchar(255)
magnesio	int(4)	genere	varchar(15)	nome	varchar(45)
calcio	int(3)	nome_vasca	varchar(45)	cognome	varchar(45)
kh	int(2)	tipo	varchar(10)	tipo	varchar(45)
ultimo_cambio_acqua	date	numero	int(4)	numeroTelefonico	varchar(20)
litri	int(7)			cambioPassword	tinyint(1)
				password	varchar(255)

Durante l'implementazione il mio data base è cambiato abbastanza rispetto l'implementazione infatti mi sono dimenticato alcune cose che durante l'implementazione sono saltate fuori oppure mi sono accorto di aver messo all'interno alcune cose inutili. Quindi ho aggiunto i campi nome, cognome, numero di telefono che mi permettono di avere qualche informazione in più sull'utente e ho rimosso la relazione tra utente e vasca. Nella tabella vasca non è cambiato niente mentre nella tabella abitante ho rimosso il nome e ho aggiunto il numero di abitanti per ogni specie dello stesso sesso.

## 4.3 Frontend

In questo capitolo spiegherò mostrerò come sono uscite le mie interfacce del frontend di tutto il mio sito web, cioè come il design delle pagine è uscito alla fine rispetto la progettazione, il tutto è stato implementato utilizzando Bootstrap.

### 4.3.1 Pagina login

#### Gestione Acquari Marini Benvenuto nella pagina di login

Email

Password

[Ho dimenticato la password](#)

---

© copyright Mattia Ruberto

La pagina di login è molto semplice, ci sono infatti i due input per il campo email e password, il bottone login che andrà a convalidare email e password e infine il bottone che permette all'utente di farsi inviare una nuova password di default che poi cambierà con una sua password a proprio piacimento.

### 4.3.2 Pagina nuova password

Inserisci la nuova password

Nuova password

Ripeti password

---

© copyright Mattia Ruberto

Questa è la pagina che permette di cambiare la password, semplicemente contiene due input dove viene inserita due volte la stessa password per essere sicuri di scrivere la stessa, se le due password sono uguali una volta premuto il bottone cambia il sito riporta alla pagina di login dove sarà possibile loggarsi con la nuova password.

#### 4.3.3 Pagina riassuntiva

Gestione Acquari Marini   Riassuntiva   Gestione vasche   Gestione utenti   Logout

### Pagina Riassuntiva

Nome	Magnesio	Calcio	Kh	Cambio d'acqua	Litri
vasca1	2	3	3	2019-12-09	60
vasca2	40	140	7	2018-03-11	120
vasca3	23	3	10	2017-11-11	232

© copyright Mattia Ruberto

Questa è la pagina riassuntiva, molto semplicemente questa è la prima pagina che compare dopo il login e rappresenta una tabella contenente tutte le informazioni delle vasche. Ha lo scopo appunto di riassumere tutti i dati delle vasche.

#### 4.3.4 Pagina gestione vasche

Gestione Acquari Marini   Riassuntiva   Gestione vasche   Gestione utenti   Logout

### Gestione Vasche

Aggiungi vasca

Nome	Magnesio	Calcio	Kh	Cambio d'acqua	Litri	Abitanti	Modifica	Rimuovi
vasca1	2	3	3	2019-12-09	60	<a href="#">Abitanti</a>	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>
vasca2	40	140	7	2018-03-11	120	<a href="#">Abitanti</a>	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>
vasca3	23	3	10	2017-11-11	232	<a href="#">Abitanti</a>	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>

© copyright Mattia Ruberto

Questa è la pagina gestione vasche, al suo interno è presente la tabella con al suo interno tutte le vasche presenti, è possibile aggiungere una nuova vasca, rimuovere o modificare una singola vasca e gestire gli abitanti per ogni vasca. Per l'aggiunta e la rimozione delle vasche solo gli utenti admin possono farlo.

#### 4.3.5 Pagina gestione utenti

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

### Pagina Gestione Utenti

[Aggiungi utente](#)

Email	Nome	Cognome	Tipo	Numero Telefono	CambioPassword	Modifica	Rimuovi
carlo.pezzotti@samtrevano.ch	Carlo	Pezzotti	Admin	+41 79 323 23 43	Non cambiare	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>
mattyspider590@gmail.com	Mattia	Ruberto	Admin	+41 79 323 23 43	Non cambiare	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>
rubertomattia@gmail.com	Giacomo	Pierino	User	+41 79 323 23 43	Da cambiare	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>

© copyright Mattia Ruberto

Pagina gestione utenti, praticamente uguale alla pagina gestione abitanti. È possibile aggiungere, modificare e rimuovere utente, non è possibile autoeliminarsi. A questa pagina possono accedere solo gli utenti admin.

#### 4.3.6 Pagina gestione abitanti

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

### Pagina Gestione Abitanti

Nome Vasca: vasca2

[Form gestione abitante](#)

Specie  Sesso  ▾ Tipo  ▾ Numero  [Aggiungi](#)

Nome Specie	Sesso	Tipo	Numero	Modifica	Rimuovi
Pesce_palla	F	Pesce	32	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>
Pesce_palla	M	Pesce	30	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>
Pesce_rosso	M	Pesce	22	<a href="#">Modifica</a>	<a href="#">Rimuovi</a>

Questa è la pagina a cui viene indirizzato l'utente quando preme il bottone abitanti di una vasca, in questo caso della vasca2 come si può vedere nell'immagine.

#### 4.3.7 Form gestione vasche

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

**i Aggiungi vasca**

Nome

Magnesio

Calcio

Kh

Cambio d'acqua  
 gg.mm.aaaa

Litri

**Aggiungi**

Questo è il form utilizzato per l'aggiunta e la modifica delle vasche, in questo caso è preparato per l'aggiunta, per la modifica viene utilizzato lo stesso ma vengono cambiate le azioni dei buttoni e le scritte.

#### 4.3.8 Form gestione utenti

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

**i Aggiungi utente**

Email

Nome

Cognome

Tipo

Numero Telefonico

Cambio Password

**Aggiungi**

Stessa cosa per quanto riguarda il form della gestione degli utenti, con un unico form viene gestita sia l'aggiunta che la modifica.

#### 4.4 Backend

In quest'altro capitolo invece andrò a spiegare le parti di codice più importanti che mettono insieme il mio sito.

Per struttura le classi del mio sito ho usato una struttura MVC, il che consiste nell'avere:

- Model

Queste classi contengono tutti i metodi che hanno accesso ai dati per esempio in questo caso del database.

- View

Si occupa di visualizzare i dati all'utente e permette le iterazioni fra quest'ultimo e i controller.

- Controller

Ricevono i comandi dell'utente attraverso le view e reagisce eseguendo delle operazioni anche con l'utilizzo delle classi Model che poi portano generalmente a un cambio di stato delle view.

Ho usato una struttura MVC semplicemente perché ti permette di gestire meglio tutto il codice, di dividere in classi e attribuire ad ognuna un compito seguendo così un filo logico che non ti fa fare confusione o errori perché non capisci la tua struttura e quindi ti permette di guadagnare tempo.

Per eseguire le richieste al database ho deciso di usare PDO, questo perché è stato il primo metodo che abbiamo imparato a scuola per eseguire iterazioni fra codice e database senza incorrere in qualche tipo di attacco e anche perché era quello che conoscevo meglio essendo il primo che abbiamo iniziato a mettere in pratica.

#### 4.4.1 Model

##### 4.4.1.1 MailModel

Per inviare l'email utilizzo la libreria PHPMailer che fornisce una raccolta di funzioni per crearle e inviarle. Ho scelto di usare il protocollo SMTP perché ovviamente è quello più utilizzato. Quindi in questa classe non farò altro che impostare tutte le informazioni per inviare l'email sfruttando il servizio mail di Google ed poi utilizzando sempre i metodi della libreria per inviarle.

Quindi nel costruttore setto le varie informazioni per poter inviare l'email, istanzio l'oggetto PHPMail, poi scelgo il protocollo, il debug per il protocollo, l'host che voglio utilizzare (quello di google), la porta, il sistema di criptazione, di che voglio usare l'autentificazione SMTP, poi metto l'username e la password del mio utente gmail con cui invierò l'email e imposto qua l'oggetto dell'email dato che sarà uguale per tutte.

```
public function __construct()
{
    require_once "GestioneAcquariMarini/libs/PHPMailer/PHPMailer.php";
    require_once "GestioneAcquariMarini/libs/PHPMailer/SMTP.php";
    require_once "GestioneAcquariMarini/libs/PHPMailer/Exception.php";
    $this->mail = new PHPMailer(true);
    $this->mail->isSMTP();
    $this->mail->SMTPDebug=SMTP::DEBUG_SERVER;
    $this->mail->Host = 'smtp.gmail.com';
    $this->mail->Port = 587;
    $this->mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $this->mail->SMTPAuth = true;
    $this->mail->Username = 'gestioneacquarimarini@gmail.com';
    $this->mail->Password = 'xxxxxx';
    $this->mail->setFrom('gestioneacquarimarini@gmail.com', 'Mattia Ruberto');
    $this->mail->Subject = 'Gestione Acquari Marini';
}
```

Poi per ogni email che vorrò inviare a dipendenza dell'utilizzo modificherò il contenuto, come per esempio questo metodo lo utilizzo per inviare le email all'utente quando la sua email è stata cambiata. Molto semplicemente aggiungo l'email destinataria, se ne può mettere più di una ma io la voglio inviare a solo una persona, e poi metto i contenuto, l'AltBody sarà il titolo e il body il contenuto dell'email. Poi infine tramite il metodo sendEmail la invio.

```
public function emailModifyUser($emailUser){
    $this->mail->addAddress($emailUser);
    $this->mail->AltBody = "<b>Conferma email</b>";
    $this->mail->Body = "La sua email è stata aggiornata con successo";
    return $this->sendEmail();
}
```

Il seguente metodo sfrutta il metodo send del PHPMailer per inviare l'email e se l'invio va a buon fine ritorna true altrimenti ritorna false.

```
public function sendEmail(){
    if ($this->mail->send()) {
        return true;
    } else {
        return false;
    }
}
```

#### 4.4.1.2 Database

Questa classe crea la connessione al database e permette così a tutte le altre classi di effettuare le varie richieste e quindi di inserire, modificare, eliminare e leggere i dati.

Molto semplicemente in questa classe è presente solo il costruttore e quando l'oggetto viene istanziato la connessione viene stabilita automaticamente. Viene creata la stringa di connessione al database mysql, contenente anche il nome del database, poi con il costruttore di PDO viene istanziata la connessione passandogli la stringa appena creata, l'username e la password. Poi vengono settati gli attributi per ritornare gli errori e viene disabilitato l'emulatore.

```
public function __construct($dbname)
{
    $this->dbname=$dbname;
    try{
        //creo PDO per mysql
        $dns = 'mysql:host=' . $this->host . ';dbname=' . $this->dbname;
        parent::__construct($dns, $this->user, $this->pass);
        //setto attributo per ritornare errori PDOException
        $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        //meglio disabilitare gli emulated prepared con i driver MySQL
        $this->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
    }
    //se ci sono errori li ritorno
    catch (PDOException $e){
        echo $e->getMessage();
    }
}
```

#### 4.4.1.3 ValidationFunction

La convalidazione avviene nello stesso modo di base sia per gli utenti, che per gli acquari e anche per gli abitanti, nella cartella libs è presente la classe ValidationFunction, dove sono presenti tutti i metodi di convalidazione che servono per convalidare tutti i tipi di dati che è possibile inserire all'interno dei vari input presenti sul sito web, quindi dalla convalidazione dell'email alla convalidazione del tipo di abitante.

Poi nelle classi HabitantValidation, UserValidation e HabitantValidation non si farà altro che effettuare la convalidazione con l'utilizzo delle funzioni presenti nella classe descritta precedentemente, controllare che tutti i dati sono corretti, quindi ritornare true se la convalidazione va a buon fine o false se c'è qualcosa di sbagliato la stringa contenente tutti gli errori avvenuti.

Per i valori dell'acquario usa la funzione validateInt che mi effettua il controllo dei numeri inseriti passandogli anche il minimo e il massimo.

```
public function validateInt($number, $min, $max){  
    if(is_numeric($number) && $number >= $min && $number <= $max){  
        return true;  
    }else{  
        return false;  
    }  
}
```

Per la validazione delle stringhe uso la seguente funzione, a cui passo la stringa e tramite il pattern vado a controllare se la stringa rispetta i criteri richiesti e che sia lunga minimo 1 e massimo 45.

```
public function validateString($string){  
    $validElement = $this->generalValidation($string);  
    $pattern = '/^a-zA-ZÀÁÈÉÒÓÙÙ\S*/i';  
  
    if (strlen($validElement) > 0 && strlen($validElement) <= 45 && preg_match($pattern, $validElement)) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Per gli input select ho delle funzioni che vanno a controllare che il parametro sia uno di quelli aspettati, in questo caso la costante NORMAL\_USER sta per "User" e ADMIN\_USER sta per "Admin". Per tutti gli input di questo tipo uso funzioni uguali a questa cambiando i dati con cui comparare il valore inserito.

```
public function validatePermission($permission){  
    if($permission == self::NORMAL_USER || $permission == self::ADMIN_USER){  
        return true;  
    }else{  
        return false;  
    }  
}
```

Per i valori primary key invece vado a fare un controllo in più rispetto agli altri, dove vado a verificare che un sia già presente lo stesso valore, quindi ho tre tipi di output, dato corretto, dato scorretto e dato già esistente.

Per controllare se la primary key esiste già vado a prende tutte i valori chiave nel database, converto l'array multidimensionale in uno normale e poi tramite la funzione in\_array verifico se sono già presenti dati uguali.

```
public function validateTankName($tankName)
{
    $validElement = $this->generalValidation($tankName);
    $pattern = '/^([A-Za-z0-9_-]+)$/';
    $arrayAllNameTank = $this->multidimensionalArrayToNormalArray($this->allNameTank);
    if (preg_match($pattern, $validElement) && strlen($validElement) > 0 && strlen($validElement) <= 45) {
        if (in_array(strtolower($tankName), $arrayAllNameTank)){
            return self::ALREADY_EXIST;
        }else{
            return self::CORRECT_INPUT;
        }
    }else{
        return self::INCORRECT_INPUT;
    }
}
```

La validazione della data è molto semplice, gli la passo sottoforma di stringa, faccio l=explode della data, istanzio una variabile contenente la data che gli è stata passata sottoforma di stringa, istanzio un alta variabile con la data corrente, quindi poi con il checkdate controllo se la data è reale e poi controllo che la data non sia futura.

```
public function validateDate($StringDate){
    $date_arr = explode("-", $StringDate);
    $date = new DateTime($StringDate);
    $current_date = new DateTime();
    $dateOk = false;
    if (count($date_arr) == 3 && strlen($StringDate) == 10) {
        if (checkdate($date_arr[2],$date_arr[1],$date_arr[0])) {
            if ($date <= $current_date) {
                $dateOk = true;
            }
        }
    }
    return $dateOk;
}
```

Le seguenti funzioni che ho illustrato sono quelle principali, le altre o sono molto simili a queste o cambiano di poco ma il concetto è lo stesso.

#### 4.4.1.4 HabitantValidation

##### NOTA

Questo è il metodo presente nell'HabitantValidation che effettua la validazione dell'abitante, le altre classi di validazione si comportano allo stesso modo ma ovviamente le funzioni di validazione saranno adattate ai campi che possiedono.

Come si può vedere gli vengono passati i dati all'interno dell'array, se sta avvenendo una modifica gli vengono passati anche i campi originari della specie e il genere per controllare se effettuare la convalidazione, dato che se la specie e il genere non vengono cambiati e la funzione effettua lo stesso il controllo della chiave primaria ci sarebbe un problema. Quindi ogni dato viene convalidato, gli errori vengono salvati nell'attributo stringErrors, e tramite la variabile booleana validationOK è possibile capire se tutti gli input sono corretti e se c'è ancora qualcosa da mettere apposto.

```
public function validation($habitante, $specie = null, $sex = null){  
    $validationOK = true;  
    $this->stringErrors = "";  
    if (!$this->validationFunction->validateSex($habitante[HABITANT_SEX])) {  
        $this->stringErrors = "Il genere dell'abitante è sbagliato";  
        $validationOK = false;  
    }  
    if (!$this->validationFunction->validateSpeciesHabitante($habitante[HABITANT_SPECIES])){  
        $this->stringErrors = "Il valore inserito nel campo specie è sbagliato";  
        $validationOK = false;  
    }  
    if($validationOK && ($specie != null || $sex != null) && ($habitante[HABITANT_SPECIES] != $specie ||  
$habitante[HABITANT_SEX] != $sex)){  
        if(!$this->validationFunction-  
>validatePrimaryKeysHabitante($habitante[HABITANT_SPECIES], $habitante[HABITANT_SEX])){  
            $this->stringErrors = "La specie e il genere inseriti esistono già";  
            $validationOK = false;  
        }  
        if (!$this->validationFunction->validateHabitantType($habitante[HABITANT_TYPE])){  
            $this->stringErrors .= "<br>Il tipo di abitante inserito è sbagliato";  
            $validationOK = false;  
        }  
        if (!$this->validationFunction->validateInt($habitante[HABITANT_NUMBER], 1, 10000)){  
            $this->stringErrors .= "<br>Il numero di habitanti è sbagliato";  
            $validationOK = false;  
        }  
    }  
    return $validationOK;  
}  
  
public function generalValidation($element){  
    return $this->validationFunction->generalValidation($element);  
}
```

#### 4.4.1.5 UserModel

##### NOTA

Le tre classe UserModel, TankModel e HabitanteModel sono praticamente uguali ma cambiano solamente i dati gestiti, quindi documenterò UserModel che è quella più completa delle tre e per le altre classi documenterò solamente i metodi che cambiano sostanzialmente.

Metodo che fa l'execute della query e il fetchAll del risultato ricevuto dal database per poi ritornare l'array dei dati letti. Questo metodo viene utilizzato per tutte quelle query dove non vengono utilizzati parametri nella query.

```
private function executeAndFetchStatement($select){  
    $this->statement = $this->connAccess->prepare($select);  
    $this->statement->execute();  
    $result = $this->statement->fetchAll(PDO::FETCH_ASSOC);  
    return $result;  
}
```

Questo metodo mi genera la password random da inserire quando viene aggiunto un nuovo utente o quando deve essere modificata la password perché l'utente l'ha dimenticata.

```
public function generetaRandomPassword(){  
    $alphabet =  
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890?(){}[]+*%&?!';  
    $pass = array();  
    for ($i = 0; $i < 8; $i++) {  
        $n = rand(0, strlen($alphabet) - 1);  
        $pass[] = $alphabet[$n];  
    }  
    return implode($pass);  
}
```

Metodo che mi ritorna tutti gli utenti del database, lo uso per riempire la tabella della pagina gestione utenti.

```
public function getAll(){  
    $selectUser = "select * from Utente";  
    $result = $this->executeAndFetchStatement($selectUser);  
    return $result;  
}
```

Questo metodo lo utilizzo per ricevere tutte le email di tutti gli utenti presenti sul database che poi mi serviranno la validazione dell'email del nuovo utente per verificare che non ci siano doppioni.

```
public function getAllEmail(){  
    $selectUser = "select email from utente";  
    $allEmail = $this->executeAndFetchStatement($selectUser);  
    return $allEmail;  
}
```

Questa funzione la utilizzo nel login, quando l'utente fa il login prendo l'email che utilizzo per ricavare tutte le informazioni dell'utente e quindi poter verificare la password, se deve cambiare la password e il tipo di utente.

```
public function getUserByEmail($email){  
    $selectUser = "select * from utente WHERE email=:email";  
    $this->statement = $this->connAccess->prepare($selectUser);  
    $this->statement->bindParam(':email', $email, PDO::PARAM_STR);  
    $this->statement->execute();  
    $result = $this->statement->fetchAll(PDO::FETCH_ASSOC);  
    return $result;  
}
```

Metodo che mi modifica i dati dell'utente sul database, al metodo gli viene passato l'array user con tutti i dati da aggiornare e l'email originale nel caso l'email fosse stata cambiata. Viene definita la query da effettuare sul database, viene tradotto il valore per il cambio password, dato che gili viene passato sottoforma di stringa perché nel sito non viene rappresentato da uno 0 o un 1 e poi viene fatto il bind di tutti i parametri ed infine viene eseguita l'operazione.

```
public function modify($user, $email){  
    $modifyUser = "UPDATE utente SET  
email=:email,nome==:name,cognome==:surname,tipo==:type,numeroTelefonico==:phoneNumber,cambio  
Password=:passwordChange WHERE email=:originalEmail";  
    $this->statement = $this->connAccess->prepare($modifyUser);  
    if($user['passwordChange'] == "Da cambiare"){  
        $passwordChange = 0;  
    }else{  
        $passwordChange = 1;  
    }  
    $this->statement->bindParam(':originalEmail', $email , PDO::PARAM_STR);  
    $this->statement->bindParam(':email', $user[USER_EMAIL] , PDO::PARAM_STR);  
    $this->statement->bindParam(':name', $user[USER_NAME] , PDO::PARAM_STR);  
    $this->statement->bindParam(':surname', $user[USER_SURNAME] , PDO::PARAM_STR);  
    $this->statement->bindParam(':type', $user[USER_TYPE] , PDO::PARAM_STR);  
    $this->statement->bindParam(':phoneNumber', $user[USER_PHONE_NUMBER] ,  
PDO::PARAM_STR);  
    $this->statement->bindParam(':passwordChange', $passwordChange , PDO::PARAM_STR);  
    $this->statement->execute();  
}
```

Questa funzione invece viene utilizzata per l'aggiunta di un nuovo utente sul database, alla funzione gli viene passato l'array utente con al suo interno tutti i dati da aggiungere, viene preparata la query da effettuare, tramite un controllo il dato per il cambio password viene tradotto in un 1 o 0, dato che nel form non potevo mettere 1 o 0 e nel database invece per risparmiare spazio ho messo un tinyint, poi viene fatto il bind di tutti i parametri da inserire e viene eseguita l'operazione.

```
public function add($user){  
    $addUser = "INSERT INTO utente  
(email,nome,cognome,tipo,numeroTelefonico,cambioPassword,password) VALUES (?,?,?,?,?,?,?,?)";  
    if($user['passwordChange'] == "Da cambiare"){  
        $passwordChange = 0;  
    }else{  
        $passwordChange = 1;  
    }  
    $this->statement = $this->connAccess->prepare($addUser);  
    $this->statement->bindParam(1, $user[USER_EMAIL]);  
    $this->statement->bindParam(2, $user[USER_NAME]);  
    $this->statement->bindParam(3, $user[USER_SURNAME]);  
    $this->statement->bindParam(4, $user[USER_TYPE]);  
    $this->statement->bindParam(5, $user[USER_PHONE_NUMBER]);  
    $this->statement->bindParam(6, $passwordChange);  
    $this->statement->bindParam(7, $user[USER_PASSWORD]);  
    $this->statement->execute();  
}
```

Questa funzione viene utilizzata per aggiungere la password aggiornata quando l'utente cambia la sua password, per farlo gli viene passata l'email per identificare l'utente a cui cambiare la password e la nuova password, il campo cambioPassword per verificare se l'utente deve cambiare la password viene impostato a 1 per dire che la password è stata cambiata, viene fatta la query, il bind dei parametri ed eseguita l'operazione.

```
public function insertPassword($email, $password, $valueChangePassword){  
    $changePassword = $valueChangePassword;  
    $selectAccesso = "UPDATE utente SET cambioPassword=:changePassword,  
password=:password WHERE (email=:email)";  
    $this->statement = $this->connAccess->prepare($selectAccesso);  
    $this->statement->bindParam(':password', $password, PDO::PARAM_STR);  
    $this->statement->bindParam(':email', $email, PDO::PARAM_STR);  
    $this->statement->bindParam(':changePassword', $changePassword, PDO::PARAM_STR);  
    $this->statement->execute();  
}
```

Funzione utilizzata per eliminare un'utente dal database, semplicemente gli viene passata l'email dell'utente per identificarlo e cancellarlo. Come per tutte le operazioni sul database c'è la query per eliminare l'utente, viene fatto il bind del parametro ed eseguita l'operazione.

```
public function delete($email){  
    $deleteUser = "DELETE FROM utente WHERE email=:email";  
    $this->statement = $this->connAccess->prepare($deleteUser);  
    $this->statement->bindParam(':email', $email, PDO::PARAM_STR);  
    $this->statement->execute();  
}
```

#### 4.4.2 Controller

##### NOTA

Di tutti i controller ho documentato solo i metodi principali tralasciando il costruttore dove istanzio principalmente gli oggetti model e gli index dove faccio solo il require delle view.

###### 4.4.2.1 Login

Quando nella pagina di login viene premuto il bottone “login”, il sito web richiama il metodo “logIn” che si occupa di verificare le credenziali e istanziare tutti i valori utili per il corretto funzionamento.

Il seguente metodo esegue per prima cosa un controllo iniziale per vedere che il bottone login sia stato effettivamente premuto, e che i campi non siano vuoti, poi prende dal POST l'email, quando ha l'email richiama il metodo “getUserByEmail” presente nella classe “UserModel” che va prendere i dati dell'utente con quell'email, se non è presente nessun utente con quella email verrà ritornato il messaggio di errore altrimenti viene verificata la password sul database con la password inserita nell'input, se la password è sbagliata ritorna il messaggio di errore altrimenti viene controllato il campo nel database “cambio\_password” dell'utente se è 0 il sito web lo porterà nella pagina dove potrà inserire la nuova password altrimenti se va tutto bene il sito web porterà l'utente alla pagina home, che sarebbe la pagina riassuntiva di tutte le pagine. Il messaggio di errore viene mostrato in base a una sessione in cui viene memorizzata una variabile booleana che se è true mostra l'errore.

```
public function logIn(){
    if (isset($_POST[LOGIN]) && !empty($_POST[USER_EMAIL]) &&
    !empty($_POST[USER_PASSWORD])) {
        $email = $this->generalValidation($_POST[USER_EMAIL]);
        $user = $this->userModel->getUserByEmail($email);
        if (count($user) > 0 && password_verify (
            $this->generalValidation($_POST[USER_PASSWORD]), $user[0][USER_PASSWORD])) {
            $_SESSION[AUTHENTICATION] = true;
            $_SESSION[USER_EMAIL] = $user[0][USER_EMAIL];
            $_SESSION[USER_TYPE] = $user[0][DB_USER_TYPE];
            if ($user[0][DB_USER_PASSWORD_CHANGE] == 0 ) {
                header("Location:" . URL . NEW_PASSWORD);
            } else {
                header("Location:" . URL . HOME);
            }
        }else{
            $_SESSION['errorLogin'] = true;
            header("Location:" . URL . LOGIN);
        }
    }
}
```

Questo invece è il metodo che viene richiamo quando l'utente schiaccia il bottone cambia password, viene generata una nuova password Random, la password viene criptata con il metodo password\_hash, viene validata l'email inserita dall'utente dato che quando viene schiacciato il botte esce fuori un popup dove bisogna inserire la propria email per cambiare la password, quindi viene controllato che l'email sia realmente di un utente del sito web, se l'email è valida viene fatto l'update nel database con il metodo updatePassword presente nella classe UserModel e viene inviate l'email all'utente con all'interno la nuova password, adesso gli basterà rifare il login con la nuova password che gli è stata inviate e il sito web in automatico gli permetterà di inserire una nuova password.

```
public function updatePassword($email){  
    $newPasswordGenerate = $this->userModel->generetaRandomPassword();  
    $passwordHash = password_hash($newPasswordGenerate, PASSWORD_DEFAULT);  
    if($this->userValidation->validateEmail($email)){  
        $this->userModel->updatePassword($email, $passwordHash);  
        $this->mailModel->emailUpdatePassword($email, $newPasswordGenerate);  
        $_SESSION["errorRequestNewPassword"] = 1;  
    }else{  
        $_SESSION["errorRequestNewPassword"] = 2;  
    }  
    header("Location:".$URL."login");  
}
```

#### 4.4.2.2 Home

Il lato backend della pagina riassuntiva è molto semplice, infatti a solo la funzione index, con il metodo getAll vengono presi tutti gli acquari presenti nel database e messi nella variabile aquariums, dopodiché viene fatto il require di tutte le view necessarie e nella view della home con la variabile aquariums verrà stampata la tabella contenente tutte le informazioni.

```
public function index(){  
    session_start();  
    if($_SESSION["authentification"] == true){  
        require_once "GestioneAcquariMarini/models/tankModel.php";  
  
        $tankManagement = new tankModel();  
        $aquariums = $tankManagement->getAll();  
  
        require "GestioneAcquariMarini/views/_templates/header.php";  
        require "GestioneAcquariMarini/views/_templates/menu.php";  
        require "GestioneAcquariMarini/views/gestioneAcquari/home/index.php";  
        require "GestioneAcquariMarini/views/_templates/footer.php";  
    }else{  
        header("Location:".$URL);  
    }  
}
```

#### 4.4.2.3 NewPassword

Il controller NewPassword possiede solo un metodo che viene utilizzato per far cambiare la password all'utente, il metodo controlla che i due input non siano vuoti e che il bottone nuova password sia stato effettivamente schiacciato, poi controlla se le due password inserite sono uguali, se sono uguali inserisce la nuova password nel database altrimenti ritorna un errore.

```
public function changePassword(){
    if (isset($_POST['submitNewPassword']) && !empty($_POST['newPassword']) &&
!empty($_POST['againNewPassword'])) {
        if ($_POST['newPassword'] == $_POST['againNewPassword']) {
            $email = $this->generalValidation($_SESSION["email"]);
            $password = password_hash($this->generalValidation($_POST['newPassword']),
PASSWORD_DEFAULT);
            $this->userModel->insertPassword($email,$password,1);
            header("Location:" . URL . LOGIN);
        } else {
            header("Location:" . URL . NEW_PASSWORD);
        }
    }
}
```

#### 4.4.2.4 TankManagement

Il seguente metodo non farà altro che settare tutti gli input se è necessario, sia che debba prendere i valori dal form, questo avviene quando si è provato ad aggiungere un acquario ma sono stati inseriti dei valori sbagliati sia che viene premuto il bottone modifica per uno specifico acquario nella tabella e quindi bisogna prendere i valori e spostarli nel form. Le prime righe servono a impostare i vari testi nei bottoni e le azioni da eseguire dato che viene usato lo stesso form per aggiungere e modificare l'acquario. Il nome dell'acquario e la stringa contenente gli errori sono opzionali infatti gli vengono passati nel caso del nome dell'acquario solo quando sta avvenendo una modifica e nel caso della stringa contenente gli errori solo quando si è già provato ad aggiungere o modificare la vasca. Dopo aver settato tutte le variabili necessarie viene fatto il require di tutte le view che servono per la visualizzazione del form.

```
public function requirePageForm($pageInformation){  
    if ($_SESSION["authentication"]){
        $title = $pageInformation[PAGE_TITLE];
        $nameButton = $pageInformation[NAME_BUTTON];
        $path = $pageInformation[PATH_BUTTON];
        $name = null;
        $stringErrors = null;
        if(count($pageInformation) >= 4){
            $stringErrors = $pageInformation[ERRORS_STRING];
        }
        if(count($pageInformation) >= 5){
            $name = $pageInformation[FORM_BENCHMARK_NAME];
        }
    }  
  
    if($this->arrayTank!=null){
        $tankName = $this->arrayTank[TANK_NAME];
        $calcium = $this->arrayTank[TANK_CALCIUM];
        $magnesium = $this->arrayTank[TANK_MAGNESIUM];
        $kh = $this->arrayTank[TANK_KH];
        $waterChange = $this->arrayTank[TANK_WATER_CHANGE];
        $liter = $this->arrayTank[TANK_LITER];
    }else if($name != null){
        $tankToModify = $this->tankManagementModel->getByName($name);
        $tankName = $tankToModify[0][DB_TANK_NAME];
        $magnesium = $tankToModify[0][DB_TANK_MAGNESIUM];
        $calcium = $tankToModify[0][DB_TANK_CALCIUM];
        $kh = $tankToModify[0][DB_TANK_KH];
        $waterChange = $tankToModify[0][DB_TANK_WATER_CHANGE];
        $liter = $tankToModify[0][DB_TANK_LITER];
    }  
    require "GestioneAcquariMarini/views/_templates/header.php";
    require "GestioneAcquariMarini/views/_templates/menu.php";
    require "GestioneAcquariMarini/views/gestioneAcquari/tank/tankManagement.php";
    require "GestioneAcquariMarini/views/_templates/footer.php";
}  
}  
}  
}
```

Quando l'utente clicca il bottone aggiungi, viene eseguito questo metodo formAddTank, che prepara il form per l'aggiunta della vasca e con poi viene richiamato il metodo requirePageForm, che è sempre presente nella stessa classe controller TankManagement.

```
public function formAddTank(){
    if($_SESSION["type"] == "Admin") {
        $path=URL."tankManagement/addTank";
        $pageInformation = array("Aggiungi vasca", "Aggiungi", $path);
        $this->requirePageForm($pageInformation);
    }
}
```

Adesso è presente il form per l'aggiunta della vasca quindi quando verrà premuto il bottone per aggiungere la vasca verrà richiamato il metodo addTank.

Il seguente metodo richiama il metodo getTankArray che ritorna tutti i valori inseriti nell'input, i valori vengono validati tramite il metodo validation, presente nella classe model TankValidation, se la validazione è corretta viene aggiunto l'acquario nel database tramite il metodo addTank presente nella classe model TankModel, l'array con i valori degli input viene settato a null e l'utente viene riportato alla pagina gestione vasche. Altrimenti se la convalidazione non va a buon fine viene richiamato il metodo requirePageForm a cui gli vengono passati anche la stringa contenente gli errori trovati durante la validazione.

```
public function addTank(){
    if($_SESSION["type"] == "Admin") {
        $this->arrayTank = $this->getTankArray();
        if ($this->tankValidationModel->validation($this->arrayTank, null)) {
            $this->tankManagementModel->add($this->arrayTank);
            $this->arrayTank = null;
            if($_SESSION["authentification"]){
                header("Location:" . URL . "tankManagement");
            }else{
                header("Location:".URL);
            }
        } else {
            $stringErrors = $this->tankValidationModel->stringErrors;
            $path = URL . "tankManagement/addTank";
            $pageInformation = array("Aggiungi vasca", "Aggiungi", $path, $stringErrors);
            $this->requirePageForm($pageInformation);
        }
    } else{
        header("Location:" . URL . HOME);
    }
}
```

Questo metodo cancella un acquario, gli viene passato come parametro il nome dell'acquario e tramite il metodo delete lo cancella.

```
public function delete($bowl){
    if($_SESSION["authentification"]){
        $this->tankManagementModel->delete($bowl);
        header("Location:" . URL . "tankManagement");
    } else{
        header("Location:".URL);
    }
}
```

Il seguente metodo seguente metodo viene eseguito quando viene schiacciato il bottone modifica nella tabella delle vasche presente nella pagina gestione vasche, si comporta come quello per l'aggiunta con la

differenza che possono farlo tutti gli utenti, quindi vengono settate tutti i dati che il form necessita quando bisogna modificare una vasca e poi con il metodo requirePageForm vengono visualizzate le view necessarie con all'interno i dati della vasca da modificare come spiegato in precedenza.

```
public function formModifyTank($name){  
    $path = URL . "tankManagement/modifyTank/".$name;  
    $pageInformation = array("Modifica vasca", "Modifica", $path, $name);  
    $this->requirePageForm($pageInformation);  
}
```

Quando nel form per la modifica viene schiacciato il bottone modifica viene richiamato il metodo modifyTank. Il seguente metodo come per l'aggiunta prende i valori che sono negli input, esegue la validazione dei valori, se i valori sono corretti esegue la modifica della vasca con i valori inseriti altrimenti viene visualizzato il form contenente i valori modificati e gli errori trovati durante la validazione.

```
public function modifyTank($name){  
    $this->arrayTank = $this->getTankArray();  
    if($this->tankValidatioModel->validation($this->arrayTank, $name)){  
        $this->tankManagementModel->modify($this->arrayTank, $name);  
        $this->arrayTank = null;  
        if($_SESSION["authentification"]){  
            header("Location:" . URL . "tankManagement");  
        }else{  
            header("Location:".URL);  
        }  
    }else{  
        $path = URL . "tankManagement/modifyTank/".$name;  
        $stringErrors = $this->tankValidatioModel->stringErrors;  
        $pageInformation = array("Modifica vasca", "Modifica", $path, $stringErrors, $name);  
        $this->requirePageForm($pageInformation);  
    }  
}
```

#### 4.4.2.5 UserManagement

##### NOTA

Il principio di funzionamento dello UserManagement è lo stesso del TankManagement, cambiano poche cose rispetto all'altra classe quindi commenterò solamente le funzioni in cui ci sono delle differenze sostanziali mentre in quelli dove cambiano solo i dati utilizzati non li ho documentati.

Il metodo che aggiunge l'utente nel database ha anche questo lo stesso funzionamento con l'unica differenza che prima di aggiungere il nuovo utente viene fatta la validazione, se la validazione va a buon fine viene inviata l'email all'utente contenente la nuova password e con la conferma dell'aggiunta dell'email, se l'email viene inviata senza problemi viene aggiunto anche l'utente sul database altrimenti viene ritornato un errore che dice che l'email non esiste.

```
public function addUser(){
    $this->arrayUser = $this->getUserArray();
    $stringErrors = "";
    if($this->userValidationModel->validation($this->arrayUser, null)){
        if($this->mailModel->emailNewUser($this->arrayUser[USER_EMAIL], $this->arrayUser[USER_PASSWORD], $this->arrayUser[USER_NAME], $this->arrayUser[USER_SURNAME])){
            $this->arrayUser[USER_PASSWORD] = password_hash($this->arrayUser[USER_PASSWORD], PASSWORD_DEFAULT);
            $this->usersManagementModel->add($this->arrayUser);
            $this->arrayUser = null;
            if($_SESSION["authentification"] == true){
                header("Location: " . URL . "userManagement");
            }else{
                header("Location:" . URL);
            }
        }
        $stringErrors .= "L'email non esiste<br>";
    }
    $stringErrors .= $this->userValidationModel->stringErrors;
    $path = URL."userManagement/addUser";
    $pageInformation = array("Aggiungi utente", "Aggiungi", $path, $stringErrors);
    $this->requirePageForm($pageInformation);
}
```

La seguente funzione effettua la modifica dei dati dell'utente, come sempre i dati inseriti vengono validati, se la validazione non va a buon fine ritorna gli errori trovati altrimenti viene effettuato un controllo che controlla se l'email è cambiata, se l'email è cambiata si invia un'email di conferma all'utente confermando che l'email è stata cambiata, se non è possibile inviare l'email viene ritornato l'errore mentre se l'email viene inviata correttamente vengono modificati anche i dati nel database, se invece l'email non è cambiata vengono modificati i dati e basta.

```
public function modifyUser($email){
    $stringErrors = "";
    $path = URL . "userManagement/modifyUser/" . $email;
    $this->arrayUser = $this->getUserArray();
    if($_SESSION["authentification"] == true){
        if($this->userValidationModel->validation($this->arrayUser, $email)){
            if($email != $this->arrayUser[USER_EMAIL]){
                if($this->mailModel->emailModifyUser($this->arrayUser[USER_EMAIL])){
                    $this->usersManagementModel->add($this->arrayUser);
                    $this->arrayUser = null;
                    header("Location:" . URL . "userManagement");
                }
                $stringErrors .= "L'email non esiste<br>";
            }else{
                $this->usersManagementModel->modify($this->arrayUser, $email);
                $this->arrayTank = null;
                header("Location:" . URL . "userManagement");
            }
        }
        $stringErrors .= $this->userValidationModel->stringErrors;
        $pageInformation = array("Modifica utente", "Modifica", $path, $stringErrors, $email);
        $this->requirePageForm($pageInformation);
    }else{
        header("Location:".URL);
    }
}
```

#### 4.4.2.6 HabitantManagement

Questo è il metodo che viene richiamato quando viene premuto il bottone abitanti di una vasca specifica, quando viene premuto oltre ai normali controlli viene effettuato il controllo per verificare che la vasca esista veramente per evitare che dall'url venga inserita una vasca che non esiste, se la vasca esiste viene salvato il nome della vasca in una sessione per averlo sempre sotto mano, poi vengono messi tutti gli abitanti che hanno nel database il riferimento di quella vasca, viene settato il testo per il form dato che viene utilizzato lo stesso form per la modifica e l'aggiunta, viene effettuato il controllo che verifica se l'array contenente i valori inseriti nel form è pieno, se è pieno i valori vengono ristampati nel form altrimenti no. Poi viene fatto il require di tutte le view necessarie.

```
function showTankHabitants($referenceTankName = 1){  
    $path = $this->pathForm;  
    if($_SESSION["authentification"] && strlen($referenceTankName) > 0 && $referenceTankName != 1){  
        if($this->tankValidation->validateTankName($referenceTankName)) {  
            $_SESSION["referencesTankName"] = $referenceTankName;  
            $habitants = $this->habitantModel->getAllHabitantForTank($_SESSION["referencesTankName"]);  
            $nameButton = $this->textButtonForm;  
            $stringErrors = $this->stringErrors;  
            $habitantManagement = "";  
            $habitantForm = false;  
            if($this->habitantArrayManagement != null){  
                $habitantManagement = $this->habitantArrayManagement;  
                $habitantForm = true;  
            }  
            $this->stringErrors=null;  
            $this->habitantArrayManagement = null;  
            require "GestioneAcquariMarini/views/_templates/header.php";  
            require "GestioneAcquariMarini/views/_templates/menu.php";  
            require "GestioneAcquariMarini/views/gestioneAcquari/habitants/index.php";  
            require "GestioneAcquariMarini/views/_templates/footer.php";  
        }else{  
            header("Location:".$URL);  
        }  
    }else{  
        header("Location:".$URL);  
    }  
}
```

Questo metodo aggiunge l'abitante nel database, mette in una variabile l'array di dati abitanti ricavati dal form tramite il POST, effettua la validazione dei valori, passandogli l'array dell'abitante. Poi aggiunge l'array nel database tramite la funzione add della classe model HabitantModel e svuota l'array. Ritorna così la pagina aggiornata.

```
public function addHabitant(){
    $this->habitantArrayManagement = $this->getArrayHabitantByPost();
    if ($this->habitantValidation->validation($this->habitantArrayManagement)) {
        $this->habitantModel->add($this->habitantArrayManagement);
        $this->habitantArrayManagement = null;
        if($_SESSION["authentification"]){
            header("Location:" . URL . "habitantManagement/showTankHabitants/" . $_SESSION["referencesTankName"]);
        }else{
            header("location: " . URL);
        }
    }else{
        $this->stringErrors = $this->habitantValidation->stringErrors;
        $this->showTankHabitants($_SESSION["referencesTankName"]);
    }
}
```

Metodo che viene eseguito quando nella tabella degli abitanti viene schiacciato il bottone modifica, alla funzione gli vengono passati la specie dell'abitante e il sesso per poterlo recuperare dal database con il metodo getHabitantBySpeciesAndSex, con il metodo successivo recupera da array multidimensionale un array normale come quello usato per l'aggiunta ma al posto che ricavare i dati dal POST sono stati ricavati dalla richiesta del database effettuata precedentemente, questo array viene salvato nell'attributo habitantArrayManagement, poi vengono impostate le informazioni da mettere nella pagina e quindi richiamato il metodo showTankHabitants che mostrerà i valori dell'array nel form.

```
public function modifyHabitant($species, $sex){
    $habitant = $this->habitantModel->getAllHabitantBySpeciesAndSex($species, $sex);
    $this->habitantArrayManagement = $this->habitantModel->getHabitantByDatabase($habitant);
    $this->textButtonForm = "Modifica";
    $this->pathForm = URL . "habitantManagement/updateHabitant/" . $species . "/" . $sex;
    $this->showTankHabitants($_SESSION["referencesTankName"]);
}
```

Dopo che i dati sono stati spostati nel form, l'utente può effettuare le varie modifiche, quando ha finito schiaccia il bottone modifica che esegue questo metodo, il metodo prende le informazioni del form e le mette nell'array, poi dalla specie e il genere originale recupera i dati che sono sul database, effettua la validazione, questa volta al metodo gli vengono passati anche la specie e il genere che sono presenti al momento nel database, questo gli serve per capire se le chiavi primarie sono state cambiate, e quindi se effettuare i vari controlli aggiuntivi. Se la convalidazione va a buon fine viene eseguita la modifica sul database, per identificare l'abitante da modificare gli vengono passati anche la specie e il genere, il nome della vasca non gli viene passato perché memorizzato nella sessione. Poi viene mostrato di nuovo il sito web, se la convalidazione è corretta la tabella sarà modificata altrimenti la tabella non avrà avuto cambiamenti, il form sarà ancora pieno e ci saranno gli errori riscontrati.

```
public function updateHabitant($species, $sex){  
    $this->habitantArrayManagement = $this->getArrayHabitantByPost();  
    $habitantByDatabase = $this->habitantModel->getHabitantByDatabase($this->habitantModel->getAllHabitantBySpeciesAndSex($species, $sex));  
    if ($this->habitantValidation->validation($this->habitantArrayManagement,  
    $habitantByDatabase["species"], $habitantByDatabase["sex"])) {  
        $this->habitantModel->modify($this->habitantArrayManagement, $habitantByDatabase["species"],  
    $habitantByDatabase["sex"]);  
        if($_SESSION["authentification"]){  
            header("Location:" . URL .  
            "habitantManagement/showTankHabitants/" . $_SESSION["referencesTankName"]);  
        }else{  
            header("location: " . URL);  
        }  
    } else {  
        $this->textButtonForm = "Modifica";  
        $this->pathForm = URL . "habitantManagement/updateHabitant/" . $species . "/" . $sex;  
    }  
    $this->stringErrors = $this->habitantValidation->stringErrors;  
    $this->showTankHabitants($_SESSION["referencesTankName"]);  
}
```

Metodo che elimina l'abitante quando nella tabella viene schiacciato il bottone “rimuovi” per uno specifico di essi, tramite la specie e il sesso è possibile identificare quale bisogna eliminare e il metodo delete della classe habitantmodel viene eliminato.

```
public function delete($species, $sex)  
{  
    if ($_SESSION["authentification"]) {  
        $this->habitantModel->delete($species, $sex);  
        header("Location:" . URL . "habitantManagement/showTankHabitants/" .  
    $_SESSION["referencesTankName"]);  
    } else {  
        header("Location:" . URL);  
    }  
}
```

#### 4.4.2.7 TankValueControl

Questa classe effettua un controllo dei valori di tutte le vasche, se qualche valore è fuori dal range ottimale la funzione di controllo invierà un email all'amministratore del sito web con l'eventuale avviso. Nei requisiti di questo progetto era richiesto che ogni settimana partisse un controllo indipendente dall'utente e che effettuasse un controllo. L'unico modo per poter sfruttare le classi già implementate della mail e del database senza dover rifare tutto ma sfruttando quelli del template MVC è stato quello come in questo caso di creare un controller che effettuasse il controllo, la classe la faccio partire tramite uno script che mi apre il sito web su questo controller, per far partire il controller non bisogna essere loggati ma la classe richiede la password per poter funzionare altrimenti ti riporta al login. Lo script si farà partire poi con un utility o un software sul computer, per esempio se il servizio hosting sarà su windows si potrà tranquillamente utilizzare "Utilità di pianificazione", che è di base su questi sistemi operativi e permette di pianificare l'esecuzione di script ripetutivamente.

Con questo metodo faccio partire il javascript che mi chiude la pagina web appena aperta per effettuare il controllo dei valori.

```
public function closePage(){
    echo '<script type="text/javascript">
        var win = window.open("about:blank", "_self");
        win.close();
    </script>';
}
```

Nella funzione valueControl verifico la password che gli viene passata nell'URL, è una sicurezza minima ma meglio che niente, dato che questa password verrà utilizzata solo sul server non è accessibile a nessuno tranne che all'amministratore che la dovrebbe già sapere. Poi prendo nell'attributo aquariums ho dentro tutti gli acquari, questi li ho recuperati nel costruttore e con un foreach ciclo tutte le vasche ed effettuo il controllo. Il controllo del range lo faccio con questo semplicissimo metodo.

```
public function checkRangeValue($value, $min, $max){
    if($value >= $min && $value <= $max){
        return true;
    }else{
        return false;
}
```

Poi alla fine se tutti i valori sono nel range non faccio niente altrimenti invio un email all'amministratore, questo tramite la variabile booleana sendEmail, che se true vuol dire che devo inviarla altrimenti no.

```
if ($this->sendEmail) {
    $this->mailModel->emailWarning($name, $message);
}
```

#### Script

Lo script non farà altro che aprire il sito web su questo controller, il controller esegue tutti i controlli necessari e poi proprio il controller penserà a chiudere la pagina. Lo script è semplicissimo e serve solo a farlo partire.

```
start
http://localhost/scuola/ProgettoGestioneAcquariMarini/tankValueControl/valueControl>Password@1@1234;
```

#### 4.5 Evento diminuzione valori settimanale

Il seguente codice non farà altro che creare un evento schedulato che si ripete ogni 7 giorni, nell'evento vengono dichiarati le variabili utili per l'utilizzo, come il nome della vasca, il valore del calcio, del magnesio e de kh, viene dichiarato un cursore che prende tutte le vasche, viene aperto e poi vengono fatte passare tutte le vasche al suo interno. A ogni ciclo di vasca vengono memorizzati all'interno delle variabili i valori dell'acquario poi viene eseguita una query che setta i valori appena letti ma sottraendo 1 per il kh, 25 per il calcio e 100 per il magnesio.

```
DROP EVENT IF EXISTS updateValueTank;

DELIMITER $$$
CREATE EVENT updateValueTank ON SCHEDULE EVERY 7 DAY DO BEGIN
    DECLARE nomeVasca varchar(50);
    DECLARE calcioVasca INT;
    DECLARE magnesioVasca INT;
    DECLARE khVasca INT;

    DECLARE cur CURSOR FOR SELECT nome FROM vasca;

    OPEN cur;
    read_loop: LOOP
        FETCH NEXT FROM cur INTO nomeVasca;
        select magnesio into magnesioVasca from vasca where nome=nomeVasca;
        select calcio into calcioVasca from vasca where nome=nomeVasca;
        select kh into khVasca from vasca where nome=nomeVasca;
        UPDATE vasca SET kh=(khVasca-1),calcio=(calcioVasca-25),magnesio=(magnesioVasca-100) WHERE (nome=nomeVasca);

        END LOOP;
        CLOSE cur;
    END $$$
DELIMITER ;
```

Per permettere all'evento di essere memorizzato nell'event scheduler dato che sarà questo che periodicamente di occuperà di farlo partire prima di salvare l'evento bisogna far partire il seguente comando.  
`SET GLOBAL event_scheduler = ON;`

#### 4.6 Protocollo di test

Test Case:	TC-001	Nome:	Test generico sito web
Riferimento:	REQ-01		
Descrizione:	Testare che il sito web sia responsive		
Prerequisiti:	Avere accesso completo al sito web		
Procedura:	<ol style="list-style-type: none"> <li>1. Aprire tutte le pagine del sito con un telefono</li> <li>2. Aprire tutte le pagine del sito con un ipad</li> <li>3. Aprire tutte le pagine del sito con un computer</li> </ol>		
Risultati attesi:	<ol style="list-style-type: none"> <li>1. In tutte le pagine del sito web con qualsiasi dispositivo standards le pagine devono essere visibili in modo corretto senza distorsioni.</li> </ol>		

Test Case:	TC-002	Nome:	Pagina di login
Riferimento:	REQ-02		
Descrizione:	Testare il corretto funzionamento della pagina di login.		
Prerequisiti:	Essere nella pagina di login e avere un utente che si è già loggato, un utente che non si è mai loggato e quindi con le pagine di default.		
Procedura:	<ol style="list-style-type: none"> <li>1. Inserire nel campo email e password le credenziali dell'utente che si è già loggato.</li> <li>2. Inserire nel campo email e password le credenziali dell'utente che non si è mai loggato e quindi con una password di default.</li> <li>3. Testare se hai dimenticato la password <ul style="list-style-type: none"> <li>o Schiacciare il bottone "Ho dimenticato la password"</li> <li>o Inserire la propria email</li> </ul> </li> </ol>		
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Il sito web deve portarti alla pagina riassuntiva</li> <li>2. Il sito web deve portarti alla pagina nuova password e permetterti di cambiare la password</li> <li>3. Deve arrivarti un email sulla posta elettronica dell'email inserita contenente la nuova password, una volta che si inserisce la nuova password il sito te la farà cambiare.</li> </ol>		

Test Case:	TC-003	Nome:	Pagina gestione vasche
Riferimento:	REQ-03		
Descrizione:	Testare la pagina gestione vasche in tutte le sue funzionalità		
Prerequisiti:	Aver effettuato la pagina di login con un account amministratore e spostarsi nella pagina gestione vasche.		
Procedura:	<ol style="list-style-type: none"> <li>1. Testare l'aggiunta della vasca             <ul style="list-style-type: none"> <li>o Schiacciare il bottone "Aggiungi vasca"</li> <li>o Inserire i parametri richiesti in modo corretto e scorretto                   <ul style="list-style-type: none"> <li>o Nome, Magnesio, Calcio, kh, Cambio d'acqua e litri</li> </ul> </li> <li>o Premere il bottone "aggiungi"</li> </ul> </li> <li>2. Testare la modifica della vasca             <ul style="list-style-type: none"> <li>o Schiacciare il bottone "Modifica" di una vasca presente nella tabella</li> <li>o Modificare i dati che si vogliono modificare</li> <li>o Premere il bottone "modifica"</li> </ul> </li> <li>3. Schiacciare il bottone rimuovi di una vasca presente nella tabella</li> </ol>		
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Dopo aver schiacciato il bottone aggiungi il sito web dovrà riportarti alla pagina gestione vasche e nella tabella dovrà esserci aggiunta la vasca se i dati inseriti sono corretti altrimenti ritornerà gli errori trovati.</li> <li>2. Dopo aver schiacciato il bottone modifica il sito web dovrà riportarti alla pagina gestione vasche e nella tabella dovrà esserci la vasca con i valori modificati se i dati inseriti sono corretti altrimenti ritornerà gli errori trovati.</li> <li>3. Dopo aver schiacciato il bottone rimuovi la vasca dovrà esser stata rimossa dalla tabella</li> </ol>		

Test Case:	TC-004	Nome:	
Riferimento:	REQ-03		
Descrizione:	Testare la pagina gestione abitanti e tutte le sue funzionalità		
Prerequisiti:	Dalla pagina gestione vasche premere il bottone abitanti di una vasca presente nella tabella		
Procedura:	<ol style="list-style-type: none"> <li>4. Testare l'aggiunta dell'abitante             <ul style="list-style-type: none"> <li>o Schiacciare il bottone "Form gestione abitanti"</li> <li>o Inserire i parametri richiesti nel form in modo corretto e scorretto                   <ul style="list-style-type: none"> <li>o Specie, Genere, Tipo e il Numero.</li> </ul> </li> <li>o Premere il bottone "aggiungi"</li> </ul> </li> <li>5. Testare la modifica dell'abitante             <ul style="list-style-type: none"> <li>o Schiacciare il bottone "Modifica" di un abitante presente nella tabella</li> <li>o Modificare i dati nel form che si vogliono modificare</li> <li>o Premere il bottone "modifica"</li> </ul> </li> <li>6. Schiacciare il bottone rimuovi di un'abitante presente nella tabella</li> </ol>		
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Dopo aver schiacciato il bottone aggiungi il sito web dovrà aggiornare la tabella con il nuovo abitante o ritornare i gli errori.</li> <li>2. Dopo aver schiacciato il bottone modifica il sito web dovrà aggiornare i dati dell'abitante nella tabella.</li> <li>3. Dopo aver schiacciato il bottone rimuovi la vasca dovrà esser stata rimossa dalla tabella</li> </ol>		

Test Case:	TC-005	Nome:	Pagina gestione utenti
Riferimento:	REQ-04		
Descrizione:	Testare la pagina gestione utenti in tutte le sue funzionalità		
Prerequisiti:	Aver effettuato la pagina di login con un account amministratore e spostarsi nella pagina gestione utenti.		
Procedura:	<ol style="list-style-type: none"> <li>1. Testare l'aggiunta dell'utente             <ul style="list-style-type: none"> <li>o Schiacciare il bottone "Aggiungi utente"</li> <li>o Inserire i parametri richiesti in modo corretto e scorretto                     <ul style="list-style-type: none"> <li>o Email, Nome, Cognome, Tipo, Numero Telefonico e Cambio Password.</li> </ul> </li> <li>o Premere il bottone "aggiungi"</li> </ul> </li> <li>2. Testare la modifica dell'utente             <ul style="list-style-type: none"> <li>o Schiacciare il bottone "Modifica" di un'utente presente nella tabella                     <ul style="list-style-type: none"> <li>o Modificare i dati che si vogliono modificare</li> <li>o Premere il bottone "modifica"</li> </ul> </li> </ul> </li> <li>3. Schiacciare il bottone rimuovi di un'utente presente nella tabella</li> </ol>		
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Dopo aver schiacciato il bottone aggiungi il sito web dovrà riportarti alla pagina gestione vasche e nella tabella dovrà esserci aggiunta la vasca se i dati inseriti sono corretti altrimenti ritornerà gli errori trovati.</li> <li>2. Dopo aver schiacciato il bottone modifica il sito web dovrà riportarti alla pagina gestione vasche e nella tabella dovrà esserci la vasca con i valori modificati se i dati inseriti sono corretti altrimenti ritornerà gli errori trovati.</li> <li>3. Dopo aver schiacciato il bottone rimuovi la vasca dovrà esser stata rimossa dalla tabella. Se hai schiacciato il bottone rimuovi del tuo utente non potrai eliminarlo.</li> </ol>		

Test Case:	TC-006	Nome:	Pagina riassuntiva
Riferimento:	REQ-05		
Descrizione:	Testare le funzionalità della pagina riassuntiva.		
Prerequisiti:	Aver effettuato la pagina di login con un account amministratore.		
Procedura:	<ol style="list-style-type: none"> <li>1. Aggiungere, modificare o eliminare una vasca</li> <li>2. Testare manualmente il controllo dei dati e la <u>diminuzione</u> dei valori settimanalmente             <ul style="list-style-type: none"> <li>o Far partire da terminale lo script presente nella cartella del progetto chiama "ScriptTankValueControl"</li> </ul> </li> <li>3. Aspettare una settimana e ricontrizzare i valori oppure controllare modificare il tempo di ripetizione dello schedule nel database</li> </ol>		
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Dopo le modifiche effettuate e aver ricaricato la pagina la tabella si deve aggiornare</li> <li>2. Se ci sono valori al di fuori del range all'amministratore arriverà un email per ogni vasca con valori sbagliati con un warning</li> <li>3. I valori devono diminuiranno a seconda di quanto perde a settimana.</li> </ol>		

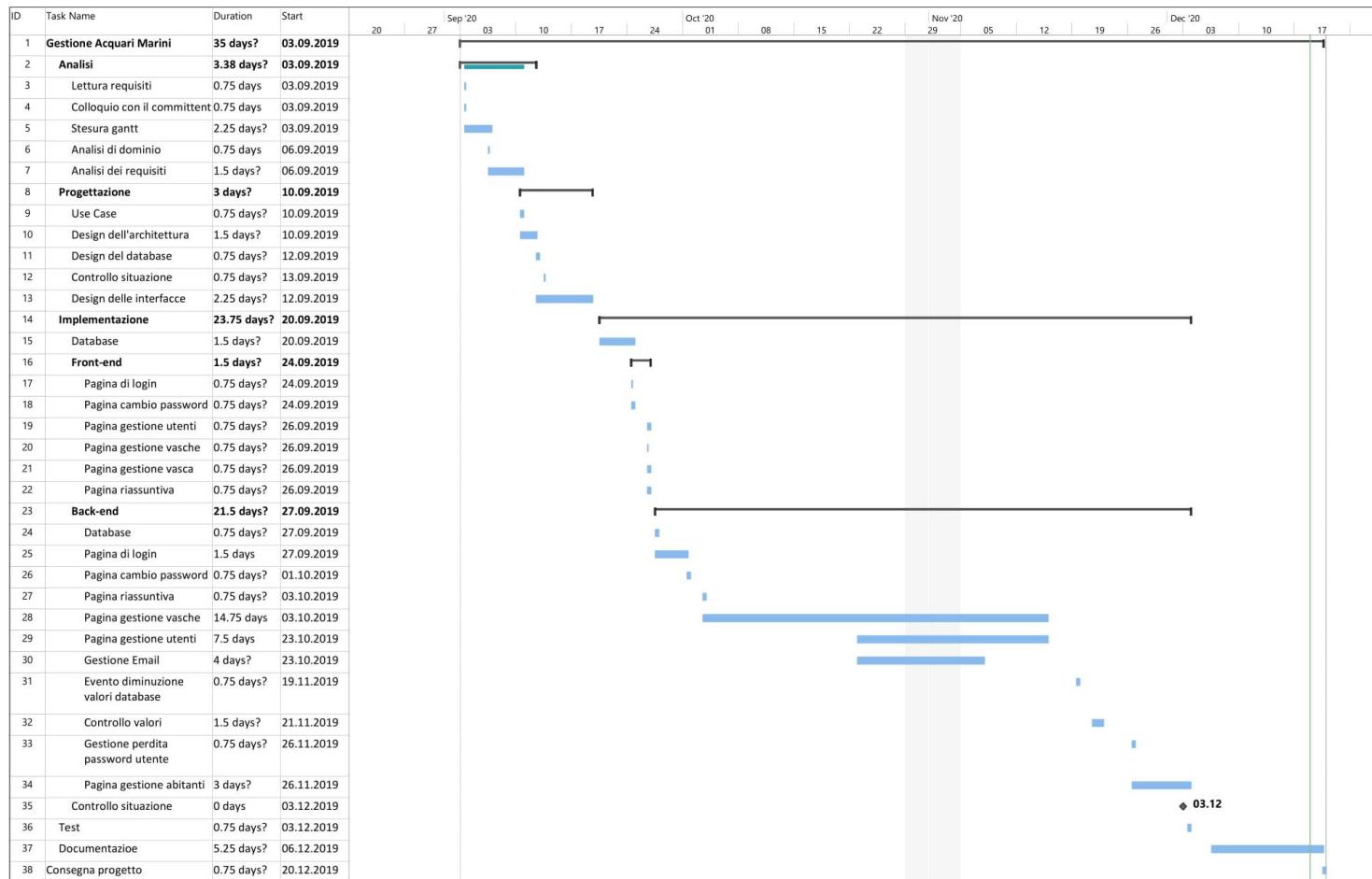
**Esempio di documentazione**

Test Case:	TC-007	Nome:	Test inserimento dati database
Riferimento:	REQ-06		
Descrizione:	Testare la sicurezza nell'inserimento dei dati		
Prerequisiti:	Avere a disposizione un qualsiasi input del sito web.		
Procedura:	<ol style="list-style-type: none"> <li>1. Testare javascript injection           <ul style="list-style-type: none"> <li>o Inserire la seguente stringa “&lt;script&gt;alert(“ciao”)&lt;/script&gt;”</li> </ul> </li> <li>2. Testare SQL injection           <ul style="list-style-type: none"> <li>o Inserire la seguente stringa “105 OR 1=1”</li> </ul> </li> </ol>		
Risultati attesi:	<ol style="list-style-type: none"> <li>2. In tutte e due i casi non deve succedere assolutamente niente</li> </ol>		

#### 4.7 Risultati test

Test Case	Nome	Descrizione	Risultato
TC-001	Test generico sito web	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-002	Pagina di login	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-003	Pagina gestione vasche	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-004	Pagina gestione abitanti	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-005	Pagina gestione utenti	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-006	Pagina riassuntiva	Verificare che la libreria e tutti i suoi metodi funzionano.	Corretto
TC-007	Test inserimento dati database	Verificare che il primo esempio sul modulo led RGB e Il potenziometro funziona correttamente.	Corretto

## 5 Consuntivo



**Titolo del progetto:** Gestione Acquari Marini  
**Alunno/a:** Mattia Ruberto  
**Classe:** I4AA  
**Anno scolastico:** 2019/2020  
**Docente responsabile:** Luca Peduzzi

Il gantt consuntivo rispetto il gantt iniziale durante la progettazione infatti per quanto riguarda l'analisi è rimasta piuttosto simile anche se i tempi un po' sono cambiati, nella progettazione invece sono partiti dagli use case, che nel primo gantt avevo messo che li avrei fatti per ultimi, ho fatto così perché essendo che non li avevo mai fatti ho preferito farli il prima possibile. Poi nell'implementazione oltre a essere cambiati i tempi è andato tutto bene fino alla pagina gestione utenti, infatti da qui ho cambiato un po' il processo, se avessi seguito la pianificazione avrei dovuto fare prima la pagine gestione vasche e dopo la pagina gestione utenti ma ho preferito fare prima questa pagina perché fra le due sembrava la più complicata e quindi poi fare quell'altra sarebbe dovuto essere più semplice. Queste due attività, pagina gestione utenti e abitanti sono andate per le lunghe a causa della convalidazione che ho fatto sia lato client che lato server ma prima avevo pensato di farla in un modo poi ho cambiato quindi ho perso un po' di tempo. Poi ho aggiunto l'attività gestione email che non avevo messo perché pensavo facesse parte della gestione utenti ma poi ho preferito metterla nel consuntivo, stessa cosa per il controllo settimanale dei valori, che ho aggiunto un'attività di cui fa parte anche l'evento per far diminuire i valori settimanalmente. Poi ho fatto la pagina gestione abitanti, anch'essa non pianificata ma riflettendoci era la soluzione più bella quella di fare una pagina dedicata agli abitanti infatti fare tutto in una pagina sarebbe stato troppo caotico. Infine ho fatto la documentazione, anche se avevo programmato di farla durante tutto l'arco del progetto alla fine l'ho fatta all'ultimo e l'integrazione dei moduli l'ho fatta strada facendo.

## 6 Conclusioni

Questo progetto potrebbe sicuramente servire a un piccolo negozio di acquari marini che gestisce ancora tutto su carta, può facilitare molto la gestione senza però essere troppo complicato e restando su funzioni basi ma utili, non penso che sia stata una perdita di tempo perché volendo si può usare questo prodotto ed è un prodotto scalabile per qualsiasi negozio infatti non è indirizzato a uno specifico negozio.

### 6.1 Sviluppi futuri

Sicuramente per coprire più mercato bisognerebbe aggiungere anche la gestione degli acquari ad acqua dolce che al momento è una limitazione non da poco di questo prodotto. Si potrebbe ottimizzare il codice e magari cambiare alcuni sistemi che sono stati sviluppati per renderlo più veloce e più bello. Il fatto che l'amministratore possa impostare direttamente una password all'utente è da implementare e magari anche la possibilità di gestire i clienti del negozio e implementare un piccolo negozio di acquari online.

### 6.2 Considerazioni personali

Sicuramente questo progetto mi è servito oltre che per rafforzare le mie conoscenze sul nuovo argomento di php fatto quest'anno per connettersi al database ma anche per riprendermi e prepararmi ai progetti futuri che dovrò fare. Ripensandoci avrei dato più tempo alla progettazione e avrei dovuto seguire un po' meglio la pianificazione fatta da me. Per il resto è stato molto utile soprattutto nel ripassare tutti i procedimenti di un progetto dato che l'ultimo progetto in terza a gruppi alcune cose non avevo dovute fare e quindi non mi ricordavo bene come dovevano essere eseguite.

<b>Titolo del progetto:</b>	Gestione Acquari Marini
<b>Alunno/a:</b>	Mattia Ruberto
<b>Classe:</b>	I4AA
<b>Anno scolastico:</b>	2019/2020
<b>Docente responsabile:</b>	Luca Peduzzi

## 7 Bibliografia

---

### 7.1 Sitografia

- <https://www.w3schools.com/>
- <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
- <http://www.sqlservertutorial.net/sql-server-triggers/sql-server-create-trigger/>
- [https://www.jetbrains.com/datagrip/features/?gclid=CjwKCAiA\\_HvBRACEiwAbViuU37Flcl2wKU2FD\\_xIDtKQMbgvyuuHaVPkB1BqN81tqfoMxEoQa4ARBoCrAEQAvD\\_BwE](https://www.jetbrains.com/datagrip/features/?gclid=CjwKCAiA_HvBRACEiwAbViuU37Flcl2wKU2FD_xIDtKQMbgvyuuHaVPkB1BqN81tqfoMxEoQa4ARBoCrAEQAvD_BwE)
- <http://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/>
- <https://dev.mysql.com/doc/refman/8.0/en/create-event.html>
- [https://www.jetbrains.com/datagrip/features/?gclid=CjwKCAiA\\_HvBRACEiwAbViuU2V3nKP7ommVpwIIFf2FVtITZIQC1TXMINPo5Z2qkk7hEv90EICQWxoCblQQAvD\\_BwE](https://www.jetbrains.com/datagrip/features/?gclid=CjwKCAiA_HvBRACEiwAbViuU2V3nKP7ommVpwIIFf2FVtITZIQC1TXMINPo5Z2qkk7hEv90EICQWxoCblQQAvD_BwE)
- <https://docs.microsoft.com/en-us/sql/relational-databases/event-classes/cursors-event-category?view=sql-server-ver15>
- <https://www.google.com/search?q=stack+overflow&oq=stack&aqs=chrome.1.69i57j0l3j69i60l4.1976j1j4&sourceid=chrome&ie=UTF-8>

## 8 Allegati

---

- Diari di lavoro
- Quaderno dei compiti
- Manuale per usare il sito
- Implementazione
  - <https://github.com/mattiaruberto/GestioneAcquariMarini>
- Codice database
- Codice evento

**DIARI**

## 1 INFORMAZIONI GENERALI

<b>Candidato</b>	Nome: Mattia	Cognome: Ruberto
	mattia.ruberto@samtrevano.ch	
<b>Luogo di lavoro</b>	Scuola Arti e Mestieri / CPT Trevano-Canobbio	
<b>Orientamento</b>	<input type="checkbox"/> 88601 Sviluppo di applicazioni <input checked="" type="checkbox"/> 88602 Informatica aziendale <input type="checkbox"/> 88603 Tecnica dei sistemi	
<b>Superiore professionale</b>	Nome: Luca	Cognome: Peduzzi
	luca.peduzzi@edu.ti.ch	
<b>Perito 1</b>	Nome:	Cognome:
<b>Perito 2</b>	Nome:	Cognome:
<b>Periodo</b>	<b>3 settembre 2019 – 20 dicembre 2019 (presentazioni: 7-17 gennaio 2020)</b>	
<b>Orario di lavoro</b>	Secondo orario scolastico 1° semestre	
<b>Numero di ore</b>	174	
<b>Pianificazione (in H o %)</b>	Analisi: 10% Implementazione: 50% Test: 10% Documentazione: 30%	

## 2 PROCEDURA

- Il candidato realizza il lavoro autonomamente sulla base del quaderno dei compiti ricevuto il 1 ° giorno.
- Il quaderno dei compiti è approvato dai periti. È anche presentato, commentato e discusso con il candidato. Con la sua firma, il candidato accetta il lavoro proposto.
- Il candidato ha conoscenza della scheda di valutazione prima di iniziare il lavoro.
- Il candidato è responsabile dei suoi dati.
- In caso di problemi gravi, il candidato o il superiore professionale avvertono immediatamente il perito.
- Il candidato ha la possibilità di chiedere aiuto, ma deve menzionarlo nella documentazione.
- Alla fine del tempo a disposizione per la realizzazione del LPI, il candidato deve inviare via e-mail il progetto al superiore professionale e al perito 1. In parallelo, una copia cartacea della documentazione dovrà essere fornita in duplice copia (superiore professionale e perito). Quest'ultima deve essere in tutto identica alla versione elettronica.

---

### 3 TITOLO

Gestionale per la manutenzione di acquari marini

---

### 4 HARDWARE E SOFTWARE DISPONIBILE

- 1 PC fornito dalla scuola con i tool necessari per lo svolgimento del progetto (Apache, MySql, php, ecc...).
- 1 Accesso presso l'hosting interno messo a disposizione dalla scuola per caricare il progetto.

---

### 5 PREREQUISITI

---

---

### 6 DESCRIZIONE DEL PROGETTO

Si tratta di gestire tramite applicativo via web e email le vasche di un negozio di acquari marini (valori dell'acqua e animali).

Un acquario marino richiede della manutenzione settimanale, tramite degli appositi test per l'acqua vanno misurati tre parametri essenziali per il corretto funzionamento dell'ecosistema.

I parametri da misurare sono:

Magnesio:	1200 min – 1450 max
Calcio:	350 min – 450 max
<i>Kh</i> (durezza carbonatica):	7 min – 11 max

Questi valori tendono a diminuire settimanalmente (indipendentemente dalla grandezza della vasca):

Magnesio:	100
Calcio:	25
<i>Kh</i> :	1

L'applicazione deve gestire anche i cambi d'acqua ogni 14 giorni (20% del litraggio totale)

e gli abitanti della vasca che vengono suddivisi in:

Pesci (M/F) – Crostacei (M/F) – Coralli

Requisiti:

- Bisogna prevedere un amministratore che possa accedere al sito in modo completo, quindi può creare, cancellare, modificare dei campi o altre modifiche direttamente dalla pagina web senza dovere andare in MySql;
- L'amministratore deve potere creare dei nuovi utenti inserendo l'account email come username e una password automatica random, la quale dovrà essere spedita automaticamente all'utente. Non deve essere creato un account email esterno tipo gmail o altro, ma deve essere gestito dal hosting. Al login dovrà esserci un sistema che

fa cambiare la password provvisoria. Bisogna anche prevedere la possibilità di richiedere una nuova password in caso di perdita;

- L'amministratore deve potere fare diventare in qualsiasi momento un utente amministratore o utente con solo diritti limitati (aggiornamento valori dell'acqua). Inoltre, dovrà gestire gli accessi e cancellare gli utenti. Deve esserci sempre almeno un amministratore;
- Deve essere creata una pagina di amministrazione nella quale vengano inserite dall'amministratore le vasche presenti nel negozio. Per ogni vasca va specificato il quantitativo di litri, il valore di magnesio, calcio e kh e i rispettivi abitanti della vasca: pesci, coralli crostacei
- Una pagina che presenti il riassunto giornaliero dei valori aggiornati (tenendo in considerazione i valori di diminuzione) delle vasche ed il numero di abitanti in esse presenti
- In caso di raggiungimento di un livello critico di uno o più valori dell'acqua l'applicativo deve poter avvisare per mail l'amministratore.
- In base al tempo a disposizione, nuovi requisiti possono essere inseriti nel progetto dopo discussione fra formatore e allievo.

---

## 7 RISULTATI FINALI

Il candidato è responsabile della consegna al superiore professionale e al perito:

- Una pianificazione iniziale (entro il primo giorno) / progetto di semestre entro la prima settimana.
- Una documentazione del progetto
- Un diario di lavoro
- Implementazione del progetto

---

## 8 PUNTI TECNICI SPECIFICI VALUTATI

La griglia di valutazione definisce i criteri generali secondo cui il lavoro del candidato sarà valutato (documentazione, diario, rispetto dei standard, qualità, ...).

Inoltre, il lavoro sarà valutato sui seguenti 7 punti specifici (punti da A14 a A20):

- 135 – *Documentazione DB, tabelle, ecc...*
- 237 – *Analisi della sicurezza (Applicazione Web)*
- 240 – *Sicurezza di base di dati*
- 148 – *Solidità verifica dei dati, intercettazione degli errori di inserimento*
- 193 – *Design del GUI*
- 254 – *Responsive Web Design*
- 232 – *Programmazione web professionale*

---

## 9 FIRMA

**Candidato**

Canobbio, 03.09.2018

---

**Superiore professionale**

Canobbio, 03.09.2018

---

**Perito 1**

(luogo e data)

---

**Perito 2**

(luogo e data)

---

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	03.09.2019

## Lavori svolti

Lettura di tutti i 20 quaderni dei compiti per potersi fare un'idea dei progetti, dopodiché ci sono stati assegnati i progetti, uno per ogni allievo, io deve fare un sito web che gestisce la manutenzione di acquari marini. Dopo ho letto attentamente il quaderno dei compiti, ho scritto da parte le domande sul testo che non capivo bene, ho chiesto al mio responsabile ed infine ho iniziato a fare una bozza del gantt.

## Problemi riscontrati e soluzioni adottate

GanttProject non permette di definire quante ore si lavora al giorno.

## Punto della situazione rispetto alla pianificazione

## Programma di massima per la prossima giornata di lavoro

Trovare un software sostitutivo al GanttProject e iniziare la progettazione.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	05.09.2019

## Lavori svolti

Ho sostituito GanttProject con Project, l'ho installato e ho iniziato a fare il gantt, prima di iniziare a fare il gantt ho fatto un'analisi dei requisiti su carta più approfondita della precedente che avevo già fatto per essere sicuro di avere le idee chiare sulle attività da svolgere durante il progetto. Infatti mi sono accorto di non aver capito due cose:

- L'amministratore nella creazione dell'utente quando assegna l'email essa non deve essere esterna ma deve essere gestita dall'hosting
- L'amministratore deve gestire gli accessi

Dopo ho iniziato il gantt, prima di iniziare ho dovuto riprendere manualità con il software dato che non lo usavo da un'anno e mezzo e non mi ricordavo bene tutte le funzionalità e come utilizzarle. Dopo aver ripreso la mano ho messo giù tutte le attività, ho settato il calendario in modo che lavoriamo solo il martedì, il giovedì e il venerdì, e ho iniziato a indicare tutti i tempi delle varie attività.

## Problemi riscontrati e soluzioni adottate

Ho risolto il problema riguardo ganttProject che ho sostituito con Project e poi ho avuto un'altro problema con Project che non mi aggiornava il diagramma del gantt, infatti anche se ho creato un nuovo calendario continuava a marcarmi che lavoravo tutta la settimana, questo problema però compariva solo sul diagramma o la rappresentazione grafica delle attività, alla fine ho risolto modificando il calendario standard e non creandone uno nuovo.

## Punto della situazione rispetto alla pianificazione

In linea con il programma.

## Programma di massima per la prossima giornata di lavoro

Finire il gantt e iniziare la documentazione, l'analisi dei requisiti che ho già fatto a bozza su carta ma devo metterla in digital e l'analisi del dominio.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	06.09.2019

## Lavori svolti

Ho finito il gantt dato che mi mancava di mettere le date di tutte le attività, ho aggiornato il calendario di project mettendo che il 17.09 non potrò lavorare perché sarò alla giornata informativa di militare e le vacanze dei morti dove perdiamo una settimana intera. Dopo ho iniziato a fare la documentazione, ho creato il documento e compilato le informazioni base tipo titolo del progetto, nome allievo, nome docente,... ho lo scopo del progetto, l'abstract nella documentazione non quello del foglio a parte, lo scopo del progetto, l'analisi del dominio e ho iniziato a fare anche se non ho finito l'analisi e la specifica dei requisiti. Al momento ho fatto solamente i seguenti requisiti:

- Pagina login
- Pagina amministrazione vasche
- Pagina amministrazione utenti

Dovrò aggiungere sicuramente un requisito per il database e devo ancora analizzare bene se fare altri requisiti riguardo i punti tecnici che sono presenti in fondo al quaderno dei compiti. Mi è venuto un dubbio riguardo a quale pagina nel sito web gli utenti normali possono modificare i valori delle varie vasche.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In linea con il programma.

## Programma di massima per la prossima giornata di lavoro

Finire l'analisi dei requisiti e iniziare la progettazione con il diagramma strutturale, il diagramma procedurale e di flusso.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	03.09.2019

## Lavori svolti

Lettura di tutti i 20 quaderni dei compiti per potersi fare un'idea dei progetti, dopodiché ci sono stati assegnati i progetti, uno per ogni allievo, io deve fare un sito web che gestisce la manutenzione di acquari marini. Dopo ho letto attentamente il quaderno dei compiti, ho scritto da parte le domande sul testo che non capivo bene, ho chiesto al mio responsabile ed infine ho iniziato a fare una bozza del gantt.

## Problemi riscontrati e soluzioni adottate

GanttProject non permette di definire quante ore si lavora al giorno.

## Punto della situazione rispetto alla pianificazione

## Programma di massima per la prossima giornata di lavoro

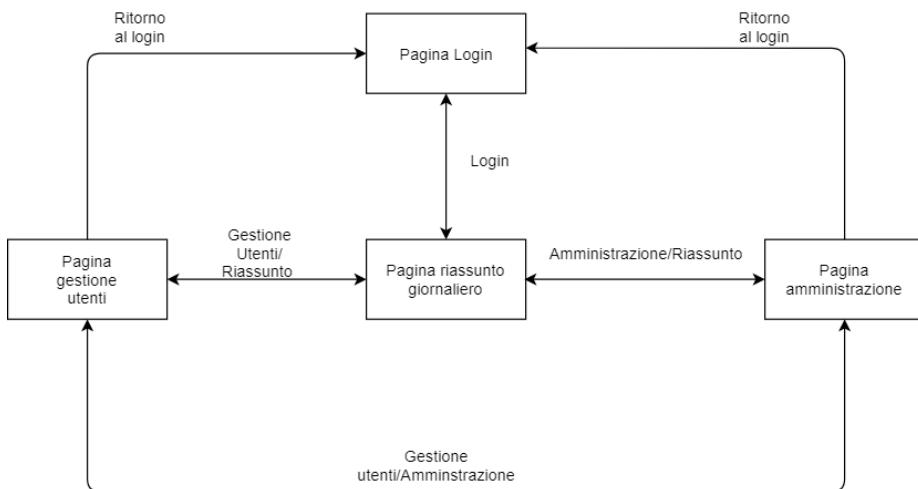
Trovare un software sostitutivo al GanttProject e iniziare la progettazione.

# Diario di lavoro

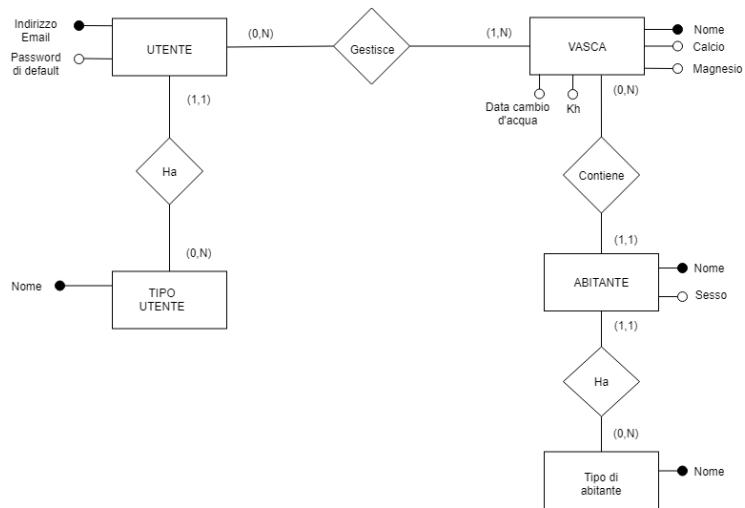
Luogo	Arti Mestieri Trevano
Data	12.09.2019

## Lavori svolti

Oggi ho fatto il design dell'architettura, non ho separato admin e utente normale perché penso che l'architettura è uguale per tutti ma cambieranno i permessi. La mia architettura parte con il login, dopo aver fatto il login ti porta alla pagina riassuntiva di tutte le vasche. Da lì si può girare per tutte le pagine se sei amministratore oppure solo per due pagine se sei un utente normale, da qualsiasi pagina si può tornare alla pagina di login.



Ho fatto il design del database:



Nel database gestisco gli utenti con una tabella per sapere il tipo di utente dato che i permessi non volevo gestirli come attributi, poi gestisco le vasche di cui voglio memorizzare per ognuna il nome, il Calcio, il magnesio e il Kh presente nell'acqua della vasca e l'ultimo cambio di acqua che è stato

effettuato, poi voglio sapere anche per ogni acquario gli abitanti presenti quindi ho ci sarà un'altra tabella dove memorizzo gli abitanti di cui voglio sapere nome e sesso e in un'altra tabella memorizzo se è un pesce, un corallo o un crostaceo.

Poi ho iniziato a fare il design delle pagine, ho finito il design della pagina di login e della pagina riassuntiva. Stavo riflettendo sul fatto che non so dove mostrare i pesci nel sito web, infatti immaginando che una vasca possa essercene molti fra pesci, crostacei e coralli stavo pensando di dedicare una pagina per tutte le informazioni dell'acquario quindi volevo sapere cosa ne pensava di lei?

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

In ritardo di 4 ore.

**Programma di massima per la prossima giornata di lavoro**

Devo finire il design di tutte le interfacce e iniziare e finire i diagrammi di flusso ed infine continuare con la documentazione.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	12.09.2019

## Lavori svolti

Oggi ho aggiunto un attributo sul litraggio della vasca nel design del database che mi sono dimenticato di aggiungere. Poi ho fatto la descrizione degli use case nella documentazione. Infine ho continuato a fare il design delle interfacce e mentre lo facevo mi sono venuti un po' di dubbi sull'architettura del sistema. Ripensando alla gestione delle vasche ho deciso che la soluzione migliore è avere una pagina generale "Gestione Vasche" che gestisce tutte le vasche presenti e ti permette di aggiungerle ed eliminarle. Per avere le informazioni della vasca si cliccherà sul nome della vasca che sarà lincato e ti porterà in una nuova pagina dove ci sarà una tabella con tutti gli abitanti presenti dove è possibile aggiungerli, modificarli ed eliminarli e un'altra tabella ancora che contiene i valori e la data dell'ultimo cambio d'acqua effettuato anche questi è possibile modificarli ma non eliminarli o crearli dato è un unico dato da salvare per ogni cosa. Poi ho pensato al login e ho deciso che quando l'utente dovrà mettere una nuova password verrà portato in una nuova pagina dove può inserire la nuova password con un bottone conferma, però su questa cosa devo ancora rifletterci perché magari potrei usare un "modal" che quando l'utente inserisce la nuova password compare e può inserire quella nuova.

Quindi ricapitolando avrò le seguenti pagine:

- Pagina Login
- Pagina nuova password (forse)
- Pagina riassuntiva
- Pagina gestione utenti
- Pagina gestione vasche
- Pagina gestione vasca

Inoltre ho notato che ho dato troppo poco tempo alla progettazione nel gantt perché pensavo che ci avrei messo di meno quindi le chiedo se devo modificarlo o devo lasciarlo così. Inoltre sempre per quanto riguarda il gantt devo modificarlo se devo aggiungere delle nuove pagine che non avevo pianificato di fare?

Oggi nel gantt avevo piazzato un milestone perché avrei dovuto incominciare a implementare ma mi sa che incomincerò a implementare venerdì prossimo come minimo dato che martedì prossimo ho militare e non ci sarò.

## Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

Programma di massima per la prossima giornata di lavoro

Finire il design delle pagine e il diagramma di flusso.

# Diario di lavoro

---

Luogo	Arti Mestieri Trevano
Data	17.09.2019

**Lavori svolti**

Oggi non ho lavorato perchè ero a militare.

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

In ritardo di 12 ore.

**Programma di massima per la prossima giornata di lavoro**

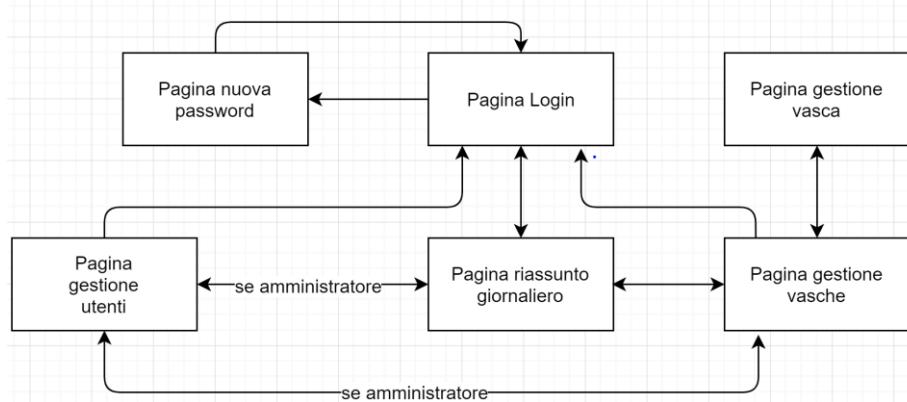
Finire il design delle pagine e il diagramma di flusso.

# Diario di lavoro

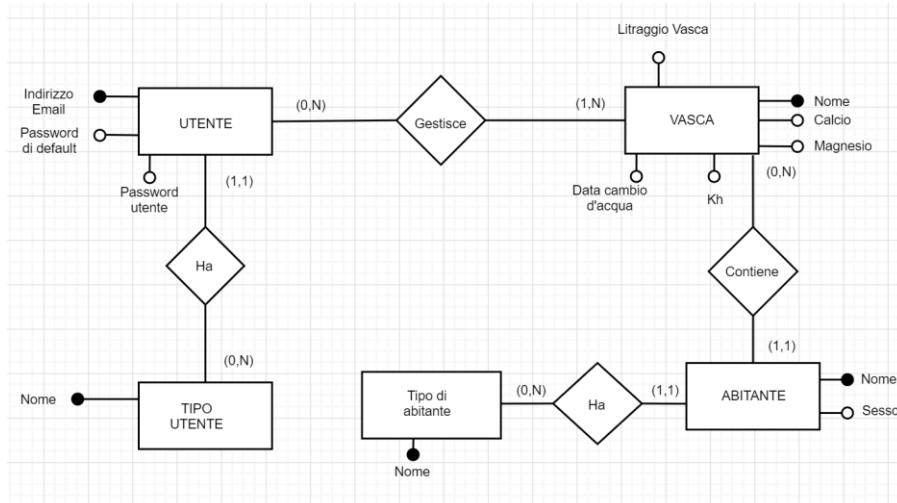
Luogo	Arti Mestieri Trevano
Data	19.09.2019

## Lavori svolti

Oggi ho sistemato definitivamente il design dell'architettura di sistema aggiungendo la pagina per inserire la nuova password e ho inserito la pagino per la gestione singola della vasca.



Ho aggiunto il campo password nella tabella utente e questa è la versione definitiva al momento del database.



Ho finito tutte le interfacce delle pagine, le ho documentate tutte come il design del database.

Queste sono le pagine principali:

La pagina riassuntiva è la prima pagina che compare dopo che l'utente ha eseguito il login se va tutto bene con le credenziali. Nella pagina riassuntiva ci sarà una tabella che rappresenta tutte le vasche con le loro informazioni sui valori dell'acqua e la data dell'ultimo cambio d'acqua inoltre avviserà anche l'utente in rosso i valori che stanno uscendo la range ideale per lo stato degli abitanti delle varie vasche.

Nella pagina gestione vache ci potranno arrivare tutti gli utenti e conterrà una tabella con il nome della vasca dove si potrà scegliere la vasca scelta e cliccandoci sopra sarà possibile aprire una nuova pagina sulla gestione personale della singola vasca. In questa pagina solo gli utenti amministratori potranno aggiungere vasche o eliminarle.

Pagina gestione vasche				
Pagina Gestione Vasca				
<b>Aggiungi abitante</b>				
<b>Nome</b>	<b>Sesso</b>	<b>Tipo</b>	<b>Modifica</b>	<b>Elimina</b>
			<input type="button" value="Button"/>	<input type="button" value="Button"/>
....	....	....	....	....
<b>Informazioni</b>				
<b>Magnesio</b>	<b>Calcio</b>	<b>Kh</b>	<b>Modifica</b>	
			<input type="button" value="Button"/>	
....	....	....	....	

La pagina gestione vasca compare dopo che nella gestione delle vasche viene cliccata una vasca specifica, a questo punto il sito ti mostra in una tabella tutti gli abitanti della vasca che possono essere pesci, corstacei e coralli. È possibile eliminarli, aggiungerli o modificarli. E in un'altra tabella ci saranno i valori dell'acqua dell'acquario e l'ultimo cambio d'acqua effettuato che potrà essere solo modificato e non eliminato o aggiunto ovviamente.

Da questa pagina permette all'amministratore di aggiungere, eliminare e modificare gli utenti che possono accedere al sito, infatti in questa pagina si potrà accedere solo con i permessi di amministratore e tutte le modifiche riguardanti gli utenti si possono fare da qui senza andare direttamente sul database. Ci sarà un tabella che rappresenta gli utenti con la loro email, password e i loro permessi. Ho iniziato poi a fare il diagramma di flusso del progetto.

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In ritardo di 16 ore.

Programma di massima per la prossima giornata di lavoro

Finire il diagramma di flusso.

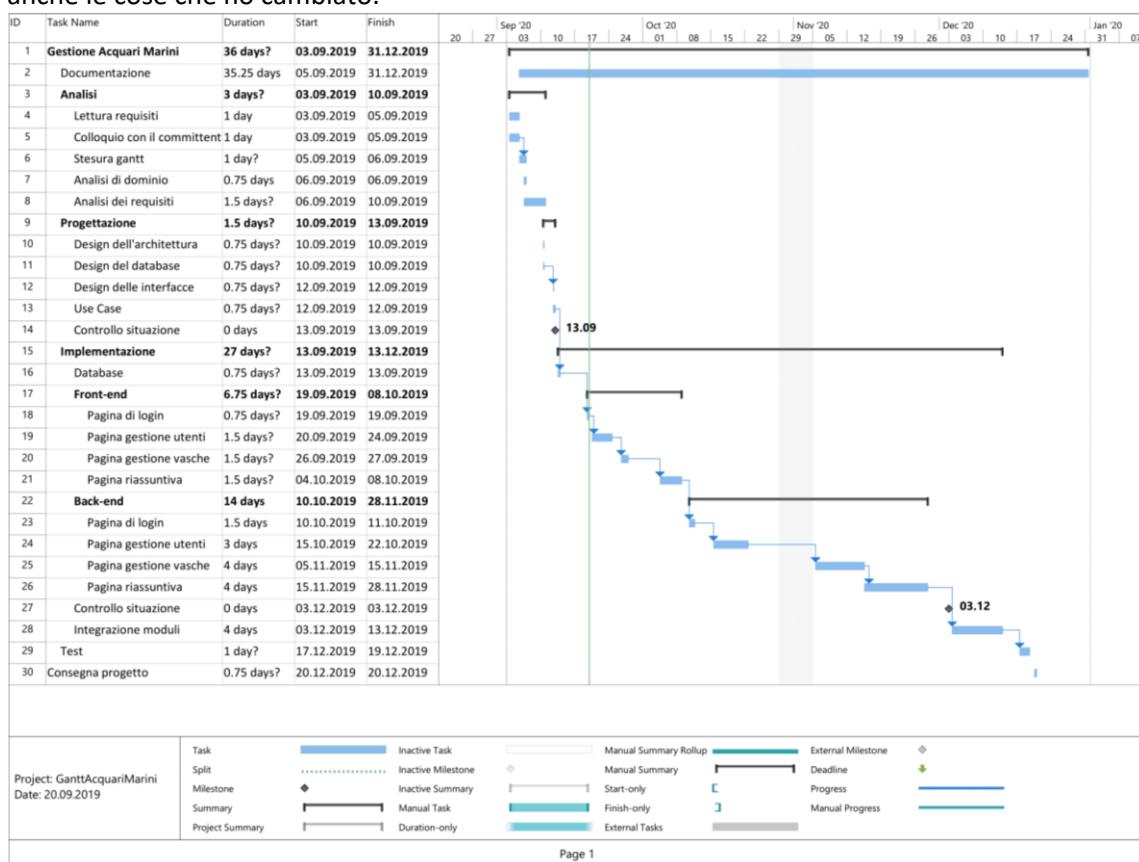
# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	20.09.2019

## Lavori svolti

Oggi come prima cosa ho chiesto al docente Valsangiacomo come dovevamo comportarci con diagramma di flusso e ci fa di fare alla fine il diagramma delle classi e di saltarlo per il momento se no sarebbe troppo lungo da fare

Poi ho cambiato due cose del gantt ma niente di fondamentale e poi l'ho documentato scrivendo anche le cose che ho cambiato.



Page 1

Le principali categorie che compongono questo gantt sono:

- Documentazione → comprende tutto ciò che metterò nella documentazione.
- Analisi → comprende tutte quelle attività che permettono di capire i requisiti e ciò che viene richiesto dal committente.
- Progettazione → tutta la parte di progettazione come design delle interfacce, design del database, use case,...
- Implementazione → tutta la parte implementativa del progetto come scrittura codice, creazione pagine, creazione database,...
- Test → Parte che comprende i test che permettono di capire se l'applicazione rispetta tutti i requisiti e fa quello che deve fare.
- Consegna progetto → fine progetto, giorno di consegna.

Nell'analisi fanno parte le seguenti categorie:

- Lettura dei requisiti → lettura del quaderno dei compiti consegnatoci dal committente in questo

caso il formatore.

- Colloquio con il committente → colloquio con il committente dove fai tutte le domande per risolvere tutti i dubbi che sono venuti fuori leggendo il quaderno dei compiti.
- Stesura gantt → pianificazione del progetto e tutte le varie tappe.
- Analisi di dominio → analisi dell'applicazione, di cosa deve fare e per chi deve farlo.
- Analisi dei requisiti → individuare tutti i requisiti che ci sono nel quaderno dei compiti.

Nella progettazione ci sono i seguenti compiti:

- Design dell'architettura → piccolo schema che riguarda la struttura dell'applicazione.
- Design del database → diagramma ER del data base.
- Design delle interfacce → design di tutte le interfacce in questo caso pagine del sito.
- Use case → diagramma uml per sintetizzare tutte le operazioni che vengono fatte dai vari utenti.

Nell'implementazione ci sono tre categorie principali:

- Frontend → implementazione di tutta la parte grafica dei vari moduli che compongono il prodotto.
- Backend → implementazione di tutta la parte logica dei vari moduli che compongono il prodotto.
- Integrazione moduli → fase in cui si prendono tutti i moduli e si integrano per andare a formare l'applicazione.

Nella fase di test viene testata l'applicazione per controllare che tutti i requisiti siano rispettati e l'applicazione faccia quello che è stato richiesto dal committente.

Ho messo due milestone, la prima l'ho piazzata prima di incominciare a implementare e la seconda prima di integrare tutti i moduli per essere sicuro di aver implementato tutto il frontend e il backend.

Questa è praticamente la prima versione di gantt infatti ho cambiato solo cambio un'attività da design procedurale a use case dato che avevo sbagliato il nome, ho tolto un milestone dato che l'avevo messa dentro a caso senza un senso e non mi convinceva più, ho tolto un'attività presentazione dato che siamo stati informati dopo che non era da inserire, e ho cambiato il nome e la data di un attività di 4 ore dato che avevo sbagliato a inserire la data. Il resto non l'ho cambiato anche se so di aver sbagliato la pianificazione della progettazione dato che mi sono dato poco tempo e nell'implementazione ho dimenticato di mettere alcune pagine che non avevo calcolato ma sono uscite fuori nella progettazione ma inserirò tutto nel consuntivo.

Poi ho iniziato a implementare, ho deciso di usare xampp per non perdere altro tempo nella configurazione di apache, php e mysql. Poi ho installato mysqlworkbench dato che preferisco lavorare con questo programma ed infine ho iniziato a implementare il database.

```
create DataBase GestioneAcquariMarini;
```

```
CREATE TABLE `gestioneacquarimarinini`.`utente` (
`email` VARCHAR(255) NOT NULL,
`passwordDefault` VARCHAR(45) NULL,
`passwordUtente` VARCHAR(45) NULL,
PRIMARY KEY (`email`));
```

```
CREATE TABLE `gestioneacquarimarinini`.`vasche` (
`Nome` NVARCHAR(45) NOT NULL,
`Calcio` DOUBLE NULL,
`Magnesio` DOUBLE NULL,
`Kh` DOUBLE NULL,
`ultimo cambio acqua` DATE NULL,
`Litraggio` INT NULL,
PRIMARY KEY (`Nome`));
```

**Problemi riscontrati e soluzioni adottate**

Mentre l'ho fatto partire non funzionava, sono andato a guardare il file di logs ma non capivo l'errore quindi ho disinstallato e rinstallato xampp, ma non funzionava ancora, poi ho cambiato la porta da 80 a 8080, poi mi diceva che vmware usava la porta 443, non ho capito perché vmware usava quella porta ma l'ho disattivato e poi funzionava.

**Punto della situazione rispetto alla pianificazione**

In ritardo di 16 ore.

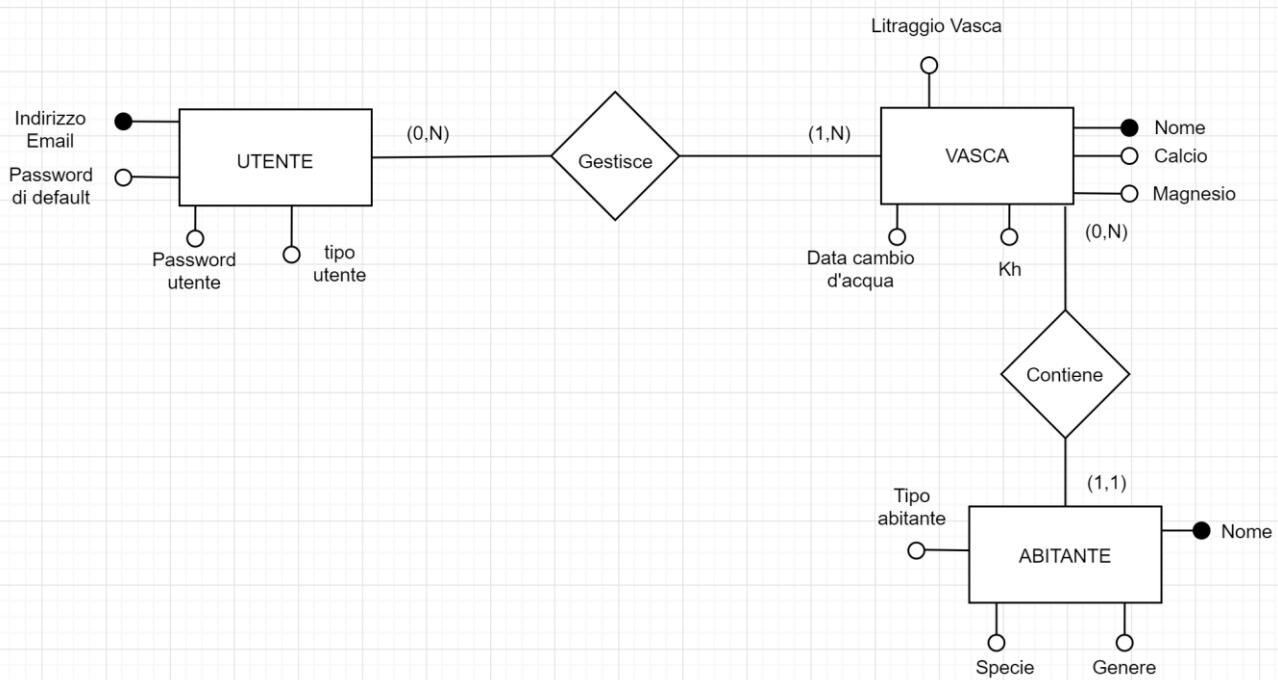
**Programma di massima per la prossima giornata di lavoro**

Finire il database e iniziare l'implementazione delle pagine.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	24.09.2019

## Lavori svolti



Per semplificare il database ho rimosso tutte quelle tabelle che avevo fatto per avere una gestione dei dati migliori perché ho pensato che avere troppe foreign key mi avrebbe complicate il lavoro per la scrittura del codice quando l'utente dovrà eliminare dei dati e per quanto riguarda l'inserimento corretto dei dati lo controllerò tramite codice.

Questo è il codice per la creazione del database:

```

drop database GestioneAcquariMarini;
create database GestioneAcquariMarini;
use GestioneAcquariMarini;

create table utente(
    email varchar(255) primary key not null,
    passwordDefault varchar(45) not null,
    tipo varchar(45) not null,
    password varchar(45)
);

create table vasca (
    nome varchar(45) primary key not null,
    calcio double,
    magnesio double,
);
  
```

```
kh double,  
`ultimo cambio acqua` date,  
Litraggio double  
);  
  
create table gestione(  
    vasca varchar(45),  
    utente varchar(255),  
    foreign key (utente) references utente(email),  
    foreign key (vasca) references vasca(nome),  
    primary key (vasca, utente)  
);  
  
create table abitante(  
    nome varchar(45) primary key not null,  
    genere varchar(45) not null,  
    tipo varchar(45) not null,  
    specie varchar(45) not null  
);
```

Poi ho fatto la struttura MVC, non sapevo se farlo o no con la struttura MVC però dato che utilizzandola dovrebbe uscire più ordinato e più semplice da capire per tutti ho deciso di usarla, questo è la struttura al momento:

- **GestioneAcquariMarini**
  - **Config**
    - **Config.php**
  - **Controllers**
    - **Login.php**
  - **Libs**
    - **Application.php**
  - **Views**
    - **\_templates**
      - **Footer.php**
      - **Header.php**
    - **gestioneAcquari**
      - **login.php**
      - **newPassword.php**
- **.htaccess**
- **Index.php**

E ho fatto la pagina di login e la pagina per la nuova password già responsive.

Pagina Login

```
<div class="containerDiv">  
    <!-- form per aggiunta strumento -->  
    <div id="login">  
        <h2>LOGIN</h2>  
        <form action=<?php echo URL; ?>login/checkLogin" method="POST">  
            <div class="form-group">  
                <label>e-mail</label>  
                <input type="text" name="email" value="" required class="form-control"/>  
            </div>  
            <div class="form-group">  
                <label>password</label>
```

```
<input type="password" name="password" value="" required class="form-control"/>
    </div>
    <input type="submit" name="submit_login" value="Login" class="btn btn-default" />
</form>
</div>
</div>
Pagina per l'inserimento della nuova password:
<div class="containerDiv">
    <!-- form per aggiunta strumento -->
    <div id="newPassword">
        <form action="<?php echo URL; ?>login/checkLogin" method="POST">
            <div class="form-group">
                <label>new password</label>
                <input type="password" name="password" value="" required class="form-control"/>
            </div>
            <div class="form-group">
                <label>repeat password</label>
                <input type="password" name="password" value="" required class="form-control"/>
            </div>
            <input type="submit" name="submit_login" value="Confirm" class="btn btn-default" />
        </form>
    </div>
</div>
```

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In ritardo di 4 ore, perché avevo calcolato che ci mettevo più tempo a fare il database e la pagina di login quindi ho recuperato 12 ore.

#### Programma di massima per la prossima giornata di lavoro

Fare il frontend della pagina gestione utenti, riassuntiva, gestione vasche, gestione vasca.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	26.09.2019

## Lavori svolti

Oggi ho finite di fare l'interfaccia di tutte le pagine, al momento tutte le pagine sono già responsive mentre alla grafica ho deciso di concentrarmi quando ho già finito tutto e non prima, ho messo tutto tranne le tabelle che verranno create in base al numero di elementi che ci saranno nel database perché mi sembra inutile stare li a metterle se poi le creerò con un ciclo. Ho deciso che la pagina sarà tutta in italiano, mentre i nomi delle classi, id e metodi saranno in inglese.

Queste sone le pagine al momento, pagina di login:

Gestione Acquari Marini  
Benvenuto nella pagina di login

e-mail

password

© copyrigth Mattia Ruberto

Pagina per l'inserimento della nuova password:

Inserisci la nuova password

nuova password

ripeti password

© copyrigth Mattia Ruberto

Pagina gestione vasca:

Gestione Acquari Marini    Gestione vasche

## Pagina Gestione Vasca

Aggiungi pesce

Specie  Numero  Sesso  ▾  
Tipi  ▾

Tabella pesci

...

Tabella valori

Magnesio	Calcio	Kh	Modifica
			Modifica

© copyrigth Mattia Ruberto

Pagina Gestione Vasche:

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Login

## Pagina Gestione Vasche

Aggiungi vasca

Nome  Litraggio

Tabella vasche

© copyrigth Mattia Ruberto

Pagina gestione utenti:

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Login

## Pagina Gestione Utenti

Gestione Utente

Email  Password  Permesso  ▾

Tabella Utenti

© copyrigth Mattia Ruberto

Come ho già detto il design al momento è di base ed è solo per permettermi di incominciare a lavorare alla parte logica del sito web, quasi sicuramente verrà modificato ora della consegna.

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

Al momento essendo che ho già fatto il frontend di tutte le pagine del sito sono in anticipo di 12 ore.

**Programma di massima per la prossima giornata di lavoro**

Iniziare la parte logica del sito web partendo dal login.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	27.09.2019

## Lavori svolti

Oggi ho fatto la classe database, che istanza la connessione con il database, ho ripreso la classe da un compito che avevo fatto per il modulo 151.

Uso il metodo PDO per far sì che se un giorno vogliono cambiare Database non devono cambiare anche la classe in questione che permette di connettersi dato che PDO è compatibile con tutti i tipi di database.

```
public function __construct($dbname)
{
    $this->dbname=$dbname;
    try{
        //creo PDO per mysql
        $dns = 'mysql:host=' . $this->host . ';dbname=' . $this->dbname;
        parent::__construct($dns, $this->user, $this->pass);
        //setto attributo per ritornare errori PDOException
        $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        // meglio disabilitare gli emulated prepared con i driver MySQL
        $this->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
    }
    //se ci sono errori li ritorno
    catch (PDOException $e){
        echo $e->getMessage();
    }
}
```

Ho fatto il controller del login

```
public function logIn()
{
    session_start();
    if (isset($_POST['login'])) {

        $username = $_POST['username'];
        $pass = hash('sha256', $_POST['pass']);

        require_once 'application/models/loginmodel.php';
        $loginModel = new LoginModel();
        $result = $loginModel->getTipo($username, $pass);

        //se trovo un elemento posso entrare
        if (count($result) > 0) {
            $_SESSION["tipo"] = $result[0]['tipo'];
            $_SESSION["nome"] = $result[0]['nome'];
            $_SESSION["cognome"] = $result[0]['cognome'];

            header("Location:" . URL . "home");
        } else {
            $this->index();
        }
    }
}
```

```
}
```

E il model del login:

```
public function getTipo($username, $pass)
{
    $selectAccesso = "select tipo,nome,cognome from accesso where username=:username
AND password=:pass";
    $this->statement = $this->connAccesso->prepare($selectAccesso);

    //inserisco i parametri
    $this->statement->bindParam(':username', $username, PDO::PARAM_STR);
    $this->statement->bindParam(':pass', $pass, PDO::PARAM_STR);

    $this->statement->execute();
    $result = $this->statement->fetchAll(PDO::FETCH_ASSOC);
    return $result;
}
```

Poi ho aggiornato Xampp e Mysql wrokbatch.

Ho perso circa 45m di lavoro perché tra le 15:00 e le 15:45 tutti quelli che sono andati in stage all'estero sono dovuti andare nell'aula 420 per un feedback con il responsabile degli stage.

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In anticipo di 8 ore.

#### Programma di massima per la prossima giornata di lavoro

Finire completamente il login testandolo e iniziando il resto.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	01.10.2019

## Lavori svolti

Oggi ho completato la pagina di login e della nuova password, al momento infatti se la password, infatti l'ultima volta avevo scritto una bozza del codice quindi c'erano dei problemi a cui non ero neanche arrivato dato che non avevo tempo.

Oggi ho risolto quei problemini, ho aggiunto il controllo per vedere se la password era stata cambiato o no semplicemente controllando se la password di default è stata settata o no. Il controllo di username e password lo faccio direttamente dalla select, infatti se non trova un utente con la stessa password e utente non ritorna niente, uso le sessioni per portarmi in giro i vari dati dell'utente, sia la scrittura sul database che la lettura è fuori pericolo dalle SQLInjection dato che uso il bind quindi non funzionano, cripto la password con sha256 al momento, ma penso che non cambierò.

Funzione di controllo del login nel controllers:

```
public function logIn(){
    if (isset($_POST['login'])) {
        $username = $_POST['email'];
        $pass = hash('sha256', $_POST['password']);
        require_once 'GestioneAcquariMarini/models/loginmodel.php';
        $loginModel = new LoginModel();
        $result = $loginModel->getUser($username, $pass);
        //se trovo un elemento posso entrare
        if (count($result) > 0) {
            session_start();
            $_SESSION["errore"] = 0;
            $_SESSION["email"] = $result[0]['email'];
            $_SESSION["password"] = $result[0]['password'];
            $_SESSION["passwordDefault"] = $result[0]['passwordDefault'];

            if($_SESSION["passwordDefault"] == null){
                header("Location:" . URL . "nuovaPassword");
            }else{
                header("Location:" . URL . "riassuntiva");
            }
        }else{
            header("Location:" . URL . "login");
        }
    }
}
```

Funzione di lettura dei dati con la funzione getUser

```
public function getUser($username, $pass)
{
    $selectAccesso = "select email,passwordDefault,password from utente WHERE
email=:email AND (password=:password OR passwordDefault=:passwordDefault)";
    $this->statement = $this->connAccesso->prepare($selectAccesso);

    //inserisco i parametri
    $this->statement->bindParam(':email', $username, PDO::PARAM_STR);
    $this->statement->bindParam(':password', $pass, PDO::PARAM_STR);
    $this->statement->bindParam(':passwordDefault', $pass, PDO::PARAM_STR);
```

```
$this->statement->execute();

$result = $this->statement->fetchAll(PDO::FETCH_ASSOC);
return $result;

}
```

Funzione nuova password:

```
public function newPassword()
{
    session_start();
    if (isset($_POST['submitNewPassword'])) {
        $password = hash('sha256', $_POST['newPassword']);
        $againPassword = hash('sha256', $_POST['againNewPassword']);

        if ($password == $againPassword) {
            require_once 'GestioneAcquariMarini/models/nuovapasswordmodel.php';
            $nuovaPasswordModel = new NuovaPasswordModel();
            $email = $_SESSION["email"];
            $nuovaPasswordModel->insertPassword($email,$password);

            header("Location:" . URL . "login");
        } else {
            $_SESSION["errorePassword"] = 1;
        }
    }
}
```

Funzione per inserire nella nuova password nel model:

```
public function insertPassword($email, $password)
{
    $selectAccesso = "UPDATE utente SET passwordDefault=:password WHERE (email
=:email)";
    $this->statement = $this->connAccesso->prepare($selectAccesso);

    //inserisco i parametri
    $this->statement->bindParam(':password', $password, PDO::PARAM_STR);
    $this->statement->bindParam(':email', $email, PDO::PARAM_STR);

    $this->statement->execute();
}
```

Devo ancora stampare i messaggi di errore riguardanti I login e Sistemare alcuni errori di sintassi.

#### Problemi riscontrati e soluzioni adottate

Ho avuto alcuni problemi con la criptografia della password e nell'utilizzo delle sessioni ma ho risolto.

#### Punto della situazione rispetto alla pianificazione

In anticipo di 8 ore.

#### Programma di massima per la prossima giornata di lavoro

Fare la pagina riassuntiva, cioè la pagina che arriva dopo il login.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	03.10.2019

## Lavori svolti

Oggi ho spostato l'html del menu in un file a parte che richiamo quando ho bisogno per non avere lo stesso codice in più file, ho creato un controller menu che mi gestisce tutti gli spostamenti che sono da fare per avere il codice il più possibile pulito e se in caso devo fare controlli nella gestione del menu posso farli da lì. Quando le credenziali sono state controllate ho aggiunto una sessione che mi permette di capire se l'utente ha fatto il login facendo un controllo ad ogni pagina in questo modo evito problemi il quale bastava sapere l'URL e si bypassare il login.

Sempre nel controller menu ho fatto la semplice funzione per il logout, in modo che prima di riportarti ai logini distrugge le sessioni.

Ho iniziato e finito la pagina riassuntiva, per cui ho aggiunto la tabella nel frontend che non avevo ancora fatto, ho creato un controller riassuntivo che tramite la classe acquario model che ho creato gestisce la tabella all'interno della pagina.

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

## Pagina Riassuntiva

Nome	Magnesio	Calcio	Kh	Cambio d'acqua	Litraggio
Vasca1	5	10	11	2019-10-03	160
Vasca2	6	1000	4	2019-08-10	1000

© copyrigth Mattia Ruberto

Poi ho iniziato a fare le pagine delle gestioni delle varie vasche, crea già la tabella con solo il nome e il litraggio della vasca, adesso devo fare in modo di poterle cancellare e aggiungere. Quando si aggiunge la vasca deve inserire solo il campo nome e i litri e non anche i valori perché ho pensato che magari un negoziante compra la vasca prima di metterla in funzione e quindi può già aggiungerla al database senza dover mettere valori a caso, però devo ancora pensarci su quest cosa.

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

## Pagina Gestione Vasche

Aggiungi vasca

Nome  Litraggio

Tabella vasche

Nome	Litraggio	Rimuovi
Vasca1	160	<input type="button" value="Rimuovi"/>
Vasca2	1000	<input type="button" value="Rimuovi"/>

© copyrigth Mattia Ruberto

**Problemi riscontrati e soluzioni adottate**

Problema di vulnerabilità che ho risolto con le sessioni a cui non avevo pensato.

**Punto della situazione rispetto alla pianificazione**

In anticipo di una settimana.

**Programma di massima per la prossima giornata di lavoro**

Fare la pagina gestione vasche, iniziare a fare la pagina gestione vasca.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	08.10.2019

## Lavori svolti

Oggi ho completato ordinato un po' la struttura MVC, aggiungendo le cartelle per le views in modo da separare i file che comprendono tutto ciò che riguarda la gestione delle vasche, la gestione degli utenti e tutto ciò che riguarda il login. Poi ho fatto il metodo per aggiungere una vasca al database, model:

```
public function add($name, $litrage){
    $addTank = "INSERT INTO gestioneacquarimarini.vasca (nome, Litraggio) VALUES (?, ?)";
    $this->statement = $this->connection->prepare($addTank);
    $this->statement->bindParam(1, $name);
    $this->statement->bindParam(2, $litrage);
    $this->statement->execute();
}

controller:
public function add(){
    if(isset($_POST[tankName]) && isset($_POST[tankLitrage])){
        if(!empty($_POST[tankName]) && !empty($_POST[tankLitrage])){
            $name = $_POST[tankName];
            $litrage = $_POST[tankLitrage];
            require_once 'GestioneAcquariMarini/models/acquarimodel.php';
            $acquariModel = new Acquarimodel();
            $acquariModel->add($name, $litrage);
            header("Location:" . URL . "gestioneVasche");
        }
    }
}
```

Fino a questo punto la mia idea era che nella pagina gestione vasche avrei avuto solo il nome e i litri della vasca, perché pensavo che almeno l'utente poteva aggiungere una vasca anche se non ancora in uso e quindi senza mettere i valori dell'acqua, ma quando ho implementato il tutto e sono arrivato ad implementare la modifica ho capito che mi sarei complicato il sito gestendo in questo modo le vasche quindi ho cambiato idea, nella pagina gestione vasche gestiro tutti gli attributi delle vasche tralasciando il fatto che una vasca potrebbe non essere in funzione infatti non gestirò questa cosa (pensavo che poteva essere una cosa in più) ma quando l'utente dovrà aggiungere una vasca dovrà mettere tutto ciò che gli riguarda, la modifica la gestisco sempre nella stessa pagina utilizzando lo stesso form anche quando l'utente andrà a modificare i valori dovrà fare così. Per la gestione dei pesci nella tabella gestione vasche per ogni vasca ci sarà un bottone abitanti che porterà a una altra pagina gestione abitanti molto simile alla pagina gestione vasche ma che riguarderà solo gli abitanti della singola vasca.

Form modifica e aggiunta vasca:

## Aggiungi vasca

Nome

Magnesio

Calcio

Kh

Cambio d'acqua  
 gg.mm.aaaa

Litri

**Aggiungi**

Quindi nel controller gestione vasche gestirò il form, modificandolo in base alle esigenze delle utente:

```
public function addTank(){
    session_start();
    if($_SESSION["authentification"] == true){
        $title = "Aggiungi vasca";
        $nameButton = "Aggiungi";

        require "GestioneAcquariMarini/views/_templates/header.php";
        require "GestioneAcquariMarini/views/_templates/menu.php";
        require
"GestioneAcquariMarini/views/gestioneAcquari/vasche/gestioneVasca.php";
        require "GestioneAcquariMarini/views/_templates/footer.php";
    }else{
        header("Location:".URL);
    }
}

public function modifyTank(){
    session_start();
    if($_SESSION["authentification"] == true){
        $title = "Modifica vasca";
        $nameButton = "Modifica";

        require "GestioneAcquariMarini/views/_templates/header.php";
        require "GestioneAcquariMarini/views/_templates/menu.php";
        require
"GestioneAcquariMarini/views/gestioneAcquari/vasche/gestioneVasca.php";
        require "GestioneAcquariMarini/views/_templates/footer.php";
    }else{
        header("Location:".URL);
    }
}
```

Problemi riscontrati e soluzioni adottate

Gestione Acquari Marini

Punto della situazione rispetto alla pianificazione
In anticipo di 8 ore.
Programma di massima per la prossima giornata di lavoro
Finire la pagina gestione vasche.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	10.10.2019

## Lavori svolti

Oggi ho implementato il metodo nel controller per aggiungere le vasche, funziona tutto devo implementare i controlli dei campi, ecco:

```
public function addTank(){
    session_start();
    if($_SESSION["authentification"] == true){
        $title = "Aggiungi vasca";
        $nameButton = "Aggiungi";
        $path = URL."gestioneVasche/addTank";

        require "GestioneAcquariMarini/views/_templates/header.php";
        require "GestioneAcquariMarini/views/_templates/menu.php";
        require
"GestioneAcquariMarini/views/gestioneAcquari/vasche/gestioneVasca.php";
        require "GestioneAcquariMarini/views/_templates/footer.php";

        if(isset($_POST["submit_management"])){
            $tank = array($_POST["tankName"], $_POST["magnesio"], $_POST["calcio"],
$_POST["kh"], $_POST["waterChange"], $_POST["liter"]);
            require_once 'GestioneAcquariMarini/models/acquarimodel.php';
            $acquariModel = new Acquarimodel();
            $acquariModel->add($tank);
        }
    }else{
        header("Location:".URL);
    }
}
```

Il metodo in acquarimodel per aggiungere l'acquaio nel database:

```
public function add($tank){
    $addTank = "INSERT INTO vasca (nome,calcio,magnesio,kh,ultimo_cambio_acqua,Litri)
VALUES (?,?,?,?,?,?)";
    $this->statement = $this->connection->prepare($addTank);
    $this->statement->bindParam(1, $tank[0]);
    $this->statement->bindParam(2, $tank[1]);
    $this->statement->bindParam(3, $tank[2]);
    $this->statement->bindParam(4, $tank[3]);
    $this->statement->bindParam(5, $tank[4]);
    $this->statement->bindParam(6, $tank[5]);
    $this->statement->execute();
}
```

Poi ho implementato sempre nel controller il metodo per gestire la modifica anche nel model degli aquari, anche questo funziona ma devo implemtare il controllo dei dati.

Controller:

```

public function modifyTank($name)
{
    session_start();
    if ($_SESSION["authentification"] == true) {
        $title = "Modifica vasca";
        $nameButton = "Modifica";
        $path = URL . "gestioneVasche/modifyTank/".$name;

        if (isset($_POST["submit_management"])) {
            $tank = array($_POST["tankName"], $_POST["calcio"], $_POST["magnesio"],
$_POST[ "kh"], $_POST["waterChange"], $_POST["liter"]);
            require_once 'GestioneAcquariMarini/models/acquarimodel.php';
            $acquariModel = new Acquarimodel();
            $acquariModel->modify($tank, $name);
        }else{
            require_once 'GestioneAcquariMarini/models/acquarimodel.php';
            $acquariModel = new Acquarimodel();

            $tankToModify = $acquariModel->getByName($name);

            $tankName = $tankToModify[0]["nome"];
            $magnesio = $tankToModify[0]["calcio"];
            $calcio = $tankToModify[0]["magnesio"];
            $kh = $tankToModify[0]["kh"];
            $waterChange = $tankToModify[0]["ultimo_cambio_acqua"];
            $liter = $tankToModify[0]["Litri"];
        }
    }

    require "GestioneAcquariMarini/views/_templates/header.php";
    require "GestioneAcquariMarini/views/_templates/menu.php";
    require
"GestioneAcquariMarini/views/gestioneAcquari/vasche/gestioneVasca.php";
    require "GestioneAcquariMarini/views/_templates/footer.php";

} else {
    header("Location:" . URL);
}

```

Model:

```

public function modify($tank, $name){
    $modifyTank = "UPDATE vasca SET nome=:tankName,
calcio=:calcio,magnesio=:magnesio,kh=:kh,ultimo_cambio_acqua=:waterChange,Litri=:liter
WHERE nome=:orginalName";
    $this->statement = $this->connection->prepare($modifyTank);
    $this->statement->bindParam(':orginalName', $name , PDO::PARAM_STR);
    $this->statement->bindParam(':tankName', $tank[0] , PDO::PARAM_STR);
    $this->statement->bindParam(':calcio', $tank[1] , PDO::PARAM_STR);
    $this->statement->bindParam(':magnesio', $tank[2] , PDO::PARAM_STR);
    $this->statement->bindParam(':kh', $tank[3] , PDO::PARAM_STR);
    $this->statement->bindParam(':waterChange', $tank[4] , PDO::PARAM_STR);
    $this->statement->bindParam(':liter', $tank[5] , PDO::PARAM_STR);
    $this->statement->execute();
}

```

Problemi riscontrati e soluzioni adottate

Piccoli problemi con l'inserimento e l'aggiornamento dei dati che però ho risolto.

Punto della situazione rispetto alla pianificazione

In anticipo di 8 ore.

Programma di massima per la prossima giornata di lavoro

Implementare i controlli dei campi e passare alla pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	11.10.2019

## Lavori svolti

Le prime due ore di lavoro sono saltate dato che il docente Valsangiacomo ci ha spiegato come funzionerà il progetto di esame e come verrà valutato quindi non ho lavorato.

Fino adesso andavo a prendere le varie librerie online, per avere una cosa più pulita le ho scaricate e messe nella struttura MVC. In questo modo anche l'head del header è più pulito dove richiamo le varie librerie.

```
<head>
    <title>Gestione Acquari Marini</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/main.css">
    <script src="js/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/validator.js"></script>
</head>
```

Nella classe validator.js ci saranno tutti i controlli in javascript.

Poi ho riciclato la classe validator da un compito del modulo 151:

```
class Validator
{
    public function validateInt($element)
    {
        $validElement = $this->generalValidation($element);
        return intval($validElement);
    }

    public function validateString($element)
    {
        $validElement = $this->generalValidation($element);

        $pattern = '/^[\w\-\_]*$/';
        if (!preg_match($pattern, $validElement)) {
            $validElement = strval($validElement);
        }
        return $validElement;
    }

    private function generalValidation($element)
    {
        $element = trim(stripslashes(htmlspecialchars($element)));
        return $element;
    }
}
```

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

In anticipo di 8 ore.

**Programma di massima per la prossima giornata di lavoro**

Implementare i controlli dei campi e passare alla pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	15.10.2019

## Lavori svolti

Oggi ho finite la pagina gestione vasche, ho completato la convalidazione sia lato client che server.

Usando queste funzioni di base:

```
function validateNumber(number){
    if (isNaN(number) || !number.toString().length > 0) {
        return false;
    } else {
        return true;
    }
}

function validateString(text){
    var reg = /^[0-9a-zA-ZÀÁÈÉÒÓÙÙ\s]+$/i;
    if (!reg.test(text) || !text.length > 0) {
        return false;
    } else {
        return true;
    }
}

function validateDate(date){
    if (date.length == 10) {
        year = date.substring(0, 4);
        month = date.substring(5, 7) - 1; // because months in JS start from 0
        day = date.substring(8, 10);

        var today = new Date();
        var dayNow = today.getDate();
        var monthNow = today.getMonth() + 1; //January is 0!
        var yearNow = today.getFullYear();

        if(year > yearNow){
            return false;
        }else if(year == yearNow){
            if(monthNow > month){
                return false;
            }else if(monthNow == month){
                if(dayNow > day){
                    return false
                }else{
                    return true;
                }
            }else{
                if(day > 0 && day < 31){
                    return true;
                }else{
                    return false;
                }
            }
        }else if(year < yearNow){
```

```
        if(month > 0 && month < 13){
            if(day > 0 && day < 31){
                return true;
            }else{
                return false;
            }
        }else{
            return false;
        }
    }
}
```

Che poi utilizzo in questo modo:

```
if(!validateString(name)){
    document.getElementById("tankName").style.border = "1px solid red";
    confirmValidate = false;
} else{
    document.getElementById("tankName").style.border = "1px solid green";
    confirmValidate = true;
}
```

Nel validator ho aggiunto una funzione che controlla se la data è corretta lato server:

```
public function validateDate($date){
    $date_arr = explode('-', $date);
    if (count($date_arr) == 3) {
        if (checkdate($date_arr[2], $date_arr[1], $date_arr[0])) {
            return $date;
        } else {
            return "0000-00-00";
        }
    } else {
        return "0000-00-00";
    }
}
```

Nel controller ho modificato il metodo addTank e modifyTank, l'ho suddiviso in due metodi e ho aggiunto una funzione getValue che ritorna un array con i valori delle vasche controllando prima i valori.

```
public function getValue(){
    $validator = new Validator();
    $name = $validator->validateString($_POST["tankName"]);
    $name = $validator->replaceSpace($name);
    $magnesio = $validator->validateInt($_POST["magnesio"]);
    $calcio = $validator->validateInt($_POST["calcio"]);
    $kh = $validator->validateInt($_POST["kh"]);
    $waterChange = $validator->validateDate($_POST["waterChange"]);
    $liter = $validator->validateInt($_POST["liter"]);

    $tank = array($name, $magnesio, $calcio, $kh, $waterChange, $liter);
    return $tank;
}
```

Per i file css e js ho messo i percorsi assoluti dal progetto.

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

In anticipo di 8 ore.

**Programma di massima per la prossima giornata di lavoro**

-Iniziare la pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	17.10.2019

## Lavori svolti

Oggi ho sistema tutti i nomi di variabili e classi che avevo scritto in italiano traducendoli in inglese per avere una certa coerenza e correttezza del codice.

Ho cambiato i controlli per il nome dell'acquario impostando che nel regex ci possono essere solamente lettere, numeri, trattino alto e basso dato che con le lettere accentate e gli spazi mi dava problemi essendo un campo primary key.

Inizialmente volevo aggiungere un campo id ma mentre lo stavo aggiungendo mi sono accorto che era meglio aggiungere dei controlli in più invece che aggiungere un id.

Controllo javascript:

```
function validatePrimaryKey(text){
    var reg = /^[0-9a-z_-]+$/i;
    if (!reg.test(text) || !text.length > 0) {
        return false;
    } else {
        return true;
    }
}
```

Controllo php:

```
public function validatePrimaryKey($element)
{
    $validElement = $this->generalValidation($element);

    $pattern = '/^[\w\-\_]*$/';
    if (!preg_match($pattern, $validElement)) {
        $validElement = strval($validElement);
    }
    return $validElement;
}
```

Ho sistemato il database modificando il nome del campo Litraggio a Litri, mettendo un varchar(255) per i campi password, ho messo i trattini al campo ultimo\_cambio\_acqua.

Riguardando il database mi sono accorto che ho la tabella gestione che dovrebbe collegare la tabella utente e la tabella vasca che non mi serve dato che non devo tenere traccia di chi crea una vasca.

Poi ho aggiunto un bottone info prima del form della aggiunta delle vasche che quando viene cliccato fa comparire tutte le informazioni sui vari campi e cosa devono contenere.

```
<input type="image" src="=PATH; ?&gt;media/img/iconInfo.png"
onclick="showInfo()"/&gt;</pre

```

Nella struttura MVC ho aggiunto la cartella media, con una sotto cartella img dove metto tutte le imagine in questo caso l'icona per le info

**Codice creazione database aggiornato:**

```
drop database GestioneAcquariMarini;
create database GestioneAcquariMarini;
use GestioneAcquariMarini;

create table utente(
    email varchar(255) primary key not null,
    passwordDefault varchar(255) not null,
    password varchar(255),
    tipo varchar(45) not null
);

create table vasca (
    nome varchar(50) primary key not null,
    calcio double,
    magnesio double,
    kh double,
    `ultimo_cambio_acqua` date,
    Litri double
);

create table gestione(
    vasca varchar(50),
    utente varchar(255),
    foreign key (utente) references utente(email),
    foreign key (vasca) references vasca(nome),
    primary key (vasca, utente)
);

create table abitante(
    nome varchar(45) primary key not null,
    genere varchar(45) not null,
    tipo varchar(45) not null,
    specie varchar(45) not null
);
```

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

In anticipo di 8 ore.

**Programma di massima per la prossima giornata di lavoro**

-Iniziare la pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	18.10.2019

## Lavori svolti

Questo è l'elenco dove viene spiegato secondo quali criteri i campi devono essere riempiti che ho modificato rispetto l'ultimo volta nello stile:

```
<ul id="listRules" class="list-group small" style="display: none;">
    <li class="list-group-item active">Regole dei campi:</li>
    <li class="list-group-item">Nome: può contenere solo lettere normali, trattino basso e alto.</li>
        <li class="list-group-item">Calcio: deve essere un numero intero.</li>
        <li class="list-group-item">Magensio: deve essere un numero intero.</li>
        <li class="list-group-item">Kh: deve essere un numero intero.</li>
        <li class="list-group-item">Cambio d'acqua: deve essere una data, la date non può essere futura.</li>
        <li class="list-group-item">Litri: deve essere un numero intero.</li>
</ul>
```

Input image:

```
<input type="image" src="<?php echo PATH; ?>media/img/iconInfo.png"
onclick="showInfo()" />
```

Funzione:

```
function showInfo(){
    var el = document.getElementById("listRules");
    if( el && el.style.display == 'none'){
        el.style.display = 'block';
    }else {
        el.style.display = 'none';
    }
}
```

Ho iniziato a fare la pagina gestione utente e ragionando sul dafarsi ho notato che devo avere un campo nome, cognome e numero di telefono, pensavo di poter identificare l'utente dall'email però pensandoci non tutti hanno un email con nome e cognome all'interno quindi devo aggiungere dei campi al database, inoltre stavo pensando se tenere il campo passwordDefault o sostituirlo con un campo boolean che se è true vuol dire che la password è stata cambiata altrimenti te la fa cambiare, sono andato per questa scelta in questo modo risparmio spazio nel database e la tabella gestione che doveva contenere le foreign key tra utente e vasca l'ho rimossa dato che non la utilizzavo.

```
use GestioneAcquariMarini;
drop table utente;

create table utente(
    email varchar(255) primary key not null,
    nome varchar(45),
    cognome varchar(45),
    tipo varchar(45),
    cambioPassword boolean,
```

```
password varchar(255)
);
```

Ho cambiato la criptazione della password in questo modo non è più decriptabile.  
`$pass = password_hash($_POST['password'], PASSWORD_DEFAULT);`

Il resto del tempo l'ho passato ad adattare il codice, ci ho messo un po' a capire come funziona.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

In anticipo di 8 ore.

Programma di massima per la prossima giornata di lavoro

-Contianuare la pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	22.10.2019

## Lavori svolti

Ho stampato correttamente la tabella gestione utenti e ho aggiunto il campo numero di telefono alla tabella utente, il campo cambio password permette al sistema di capire se l'utente deve o no cambiare password, se è uno non deve cambiarla se è 0 deve cambiarla.



© copyright Mattia Ruberto

Ho fatto il bottone rimuovi per l'utente, controller:

```
public function delete($email){  
    $this->usersManagementModel->delete($email);  
    header("Location:" . URL . "userManagement");  
}
```

model:

```
public function delete($email)  
{  
    $deleteUser = "DELETE FROM vasca WHERE email=:email";  
    $this->statement = $this->connection->prepare($deleteUser);  
    $this->statement->bindParam(':email', $email, PDO::PARAM_STR);  
    $this->statement->execute();  
}
```

Poi ho iniziato a implementare il bottone aggiungi utente, ho preso il form degli acquari e l'ho configurato per l'utente, devo ancora cambiare il tipo di input del numero telefonico:

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

### i Aggiungi utente

Email

Nome

Cognome

Tipo

Numero Telefonico  
 gg.mm.aaaa

Cambio Password

Poi ho iniziato a fare la validazione dei vari campi, funzioni in più che uso:

```

function validatePhone(numberCell){
    var reg = /^[0-9\s+\#]+$/i;
    if(numberCell.length >= 9 && numberCell.length <= 30){
        if(!reg.test(numberCell)){
            return false;
        }else{
            return true;
        }
    }else{
        return false;
    }
}

function validateString(string, characterMin, characterMax){
    var reg = /^[a-zA-ZÀÁÈÉÒÓÙÚàáèéòóùú\s]+$/i;
    if(string.length >= characterMin && string.length <= characterMax){
        if(!reg.test(string)){
            return false;
        }else{
            return true;
        }
    }else{
        return false;
    }
}

function validateEmail(email){
    var reg = /^[(([^<>()[]\.\,\;\:\s@"]+(\.\[^<>()[]\.\,\;\:\s@"]+)*|(\".+\")|((\\"[\0-9]{1,3}\.\[0-9]{1,3}\.\[0-9]{1,3}\.\[0-9]{1,3}\])|(([a-zA-Z\-\0-9]+\.)+[a-zA-Z]{2,}))$/;
    if (!reg.test(text) || !text.length > 0) {
        return false;
    } else {
        return true;
    }
}

```

Funzione completa di validazione:

```
function validateUser(){
    var confirmValidate = true;

    var email = document.getElementById("email").value;
    var name = document.getElementById("name").value;
    var surname = document.getElementById("surname").value;
    var type = document.getElementById("type").value;
    var numberCell = document.getElementById("numberCell").value;
    var passwordChange = document.getElementById("passwordChange").value;

    if(!validateEmail(email)){
        document.getElementById("email").style.border = "1px solid red";
        confirmValidate = false;
    }else{
        document.getElementById("email").style.border = "1px solid green";
    }

    if(!validateNumber(name)){
        document.getElementById("name").style.border = "1px solid red";
        confirmValidate = false;
    }else{
        document.getElementById("name").style.border = "1px solid green";
    }

    if(!validateNumber(surname)){
        document.getElementById("surname").style.border = "1px solid red";
        confirmValidate = false;
    }else{
        document.getElementById("surname").style.border = "1px solid green";
    }

    if(!validateNumber(type)){
        document.getElementById("type").style.border = "1px solid red";
        confirmValidate = false;
    }else{
        document.getElementById("type").style.border = "1px solid green";
    }

    if(!validateNumber(numberCell)){
        document.getElementById("numberCell").style.border = "1px solid red";
        confirmValidate = false;
    }else{
        document.getElementById("numberCell").style.border = "1px solid green";
    }

    if(!validateDate(passwordChange)){
        document.getElementById("passwordChange").style.border = "1px solid red";
        confirmValidate = false;
    }else{
        document.getElementById("passwordChange").style.border = "1px solid green";
    }
}

if(confirmValidate){
    document.getElementById("formAddUser").submit();
}
```

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione
In anticipo di 8 ore.
Programma di massima per la prossima giornata di lavoro -Contianuare la pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	23.10.2019

## Lavori svolti

Ho sistemato il forma per aggiungere l'utente, ho usato una select sia per il tipo di utente che per il cambio password che conterrà “cambiata” e “da cambiare”, poi a livello codice farò in modo di tradurre da quello che salverò sul database infatti su di esso continuo a salvare l'informazione in booleano. I controlli tramite javascript adesso vanno completamente devo ancora finire di adattare i controlli tramite php ma non mi prenderà tanto tempo.

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

**i Aggiungi utente**

Email  
mattia.ruberto@samtrevano.ch

Nome  
Mattia

Cognome  
Verdi

Tipo  
User

Numero Telefonico  
+79 2332 233 23

Cambio Password  
Da cambiare

**Aggiungi**

Aiutandomi con internet ho fatto la funzione che genera automaticamente una password random molto semplicemente in questo modo:

```
public function generaRandomPassword(){
    $alphabet =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890?(){}[]+*%&?!';
    $pass = array();
    $alphaLength = strlen($alphabet) - 1;
    for ($i = 0; $i < 8; $i++) {
        $n = rand(0, $alphaLength);
        $pass[] = $alphabet[$n];
    }
    return implode($pass);
}
```

Quindi all'array dell'utente che contiene tutte le informazioni riguardanti a esso ho aggiunto il campo password che mi ero dimenticato di mettere che viene tirata fuori dal metodo qua sopra.

Poi ho provato a fare una classe che mi gestisce l'email da inviare all'utente quando il suo account viene creato:

Ho trovato questa guida in cui veniva descritto come una cosa semplicissima:

[https://www.mrwebmaster.it/php/funzione-mail\\_6466.html](https://www.mrwebmaster.it/php/funzione-mail_6466.html)

Quindi ho provato a fare così:

```
public function sendEmailToNewUser($emailNewUser, $newPassword, $name, $surname){
    // definisco mittente e destinatario della mail
    $name_mittente = "Mattia Ruberto";
    $mail_mittente = "mattia.ruberto@samtrevano.ch";
    $mail_destinatario = $emailNewUser;

    // definisco il subject ed il body della mail
    $mail Oggetto = "Gestione Acquari Marini";
    $mail_Corpo = "Benvenuto ".$name." ".$surname.", il tuo account è stato registrato con successo, adesso non dovrà fare altro che accedere al sito con la password di default: \n\r".$newPassword." e cambiare la password con una tua personale, \n\rCordiali saluti, \n\r Mattia";

    // aggiusto un po' le intestazioni della mail
    // E' in questa sezione che deve essere definito il mittente (From)
    // ed altri eventuali valori come Cc, Bcc, ReplyTo e X-Mailer
    $mail_Headers = "From: " . $name_mittente . " <" . $mail_mittente . ">\r\n";
    $mail_Headers .= "Reply-To: " . $mail_mittente . "\r\n";
    $mail_Headers .= "X-Mailer: PHP/" . phpversion();

    if (mail($mail_destinatario, $mail_Oggetto, $mail_Corpo, $mail_Headers)) {
        echo "Messaggio inviato con successo a " . $mail_destinatario;
    } else {
        echo "Errore. Nessun messaggio inviato.";
    }
}

<?php

class MailModel
{
    public function sendEmailToNewUser($emailNewUser, $newPassword, $name, $surname)
    {

        require_once "PHPMailer.php";

        $mail = new PHPMailer();

        $mail->From = 'mattia.ruberto@samtrevano.ch';
        $mail->FromName = 'Mattia Ruberto';
        $mail->addAddress($emailNewUser);      // Add a recipient

        $mail->isHTML(true);                  // Set email format to HTML

        $mail->Subject = 'Gestione Acquari Marini';
        $mail->Body = "<b>Benvenuto ".$name." ".$surname."</b>";
        $mail->AltBody = "Il tuo account è stato creato con successo, adesso non dovrà fare altro che accedere al sito con la password di default: <br>" . $newPassword .
        "<br> e cambiare la password con una tua personale,<br> Cordiali saluti,<br> Mattia</p>";

        if (!$mail->send()) {
            echo 'Message could not be sent.';
        }
    }
}
```

```
        echo 'Mailer Error: ' . $mail->ErrorInfo;
    } else {
        echo 'Message has been sent';
    }
}
?>
```

Poi mi sono accorto che in questo modo non può funzionare dato che all'inizio non avevo pensato che tutti i metodi che mi venivano illustrati si basavano sul fatto che chi li utilizzava avesse un mail server, ma dato che io non voglio creare un mail server ho cercato altre soluzioni senza un mail server e ho trovato una libreria su github:

<https://github.com/PHPMailer/PHPMailer/blob/master/src/PHPMailer.php>

In teoria, dico in teoria perché non sono ancora riuscito a testarla dovrei riuscire a inviare l'email tramite il mio servizio di posta, in questo caso quello di trevano senza utilizzare un server mail sulla mia macchina.

Quindi ho caricato la libreria e ho provato ma non sono riuscito a finire l'operazione, mancano ancora alcuni passaggi che devo ben capire come per esempio il fatto di configurare il mio account email per inviare l'email dato che ancora non l'ho fatto ma non ho capito come fare:

```
require_once "PHPMailer.php";

$mail = new PHPMailer();

$mail->From = 'mattia.ruberto@samtrevano.ch';
$mail->FromName = 'Mattia Ruberto';
$mail->addAddress($emailNewUser); // Add a recipient

$mail->isHTML(true); // Set email format to HTML

$mail->Subject = 'Gestione Acquari Marini';
$mail->Body = "<b>Benvenuto ".$name." ".$surname."</b>";
$mail->AltBody = "Il tuo account è stato creato con successo, adesso non dovrà fare altro che accedere al sito con la password di default: <br>" . $newPassword . "<br> e cambiare la password con una tua personale,<br> Cordiali saluti,<br> Mattia</p>";

if (!$mail->send()) {
    echo 'Message could not be sent.';
    echo 'Mailer Error: ' . $mail->ErrorInfo;
} else {
    echo 'Message has been sent';
}
```

#### Problemi riscontrati e soluzioni adottate

Non riesco a configurare il mio account email nella class PHPMailer che mi permette di inviare le email con il mio account.

#### Punto della situazione rispetto alla pianificazione

In anticipo di 12 ore.

#### Programma di massima per la prossima giornata di lavoro

-Continuare la pagina gestione utenti, in particolare finire l'invio dell'email

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	23.10.2019

## Lavori svolti

Oggi ho creato un account email per il progetto: gestioneacquarimarin@gmail.com.  
Poi sempre da questo repository ho importato le classi SMTP.php e Exception.php dato che ieri non lo avevo fatto ed era per quello che non funzionava.

Poi ho sistemato la classe Mailodel:

Nel costruttore setto tutte le informazioni utili per connettermi al protocollo SMTP di gmail:

```
public function __construct()
{
    require_once "GestioneAcquariMarini/libs/PHPMailer/PHPMailer.php";
    require_once "GestioneAcquariMarini/libs/PHPMailer/SMTP.php";
    require_once "GestioneAcquariMarini/libs/PHPMailer/Exception.php";

    $this->mail = new PHPMailer(true);
    $this->mail->isSMTP();
    $this->mail->SMTPDebug=SMTP::DEBUG_SERVER;
    $this->mail->Host = 'smtp.gmail.com';
    $this->mail->Port = 587;
    $this->mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $this->mail->SMTPAuth = true;
    $this->mail->Username = 'gestioneacquarimarin@gmail.com';
    $this->mail->Password = 'XXXX';
}
```

Mentre in questo metodo invio l'email settando destinatario, oggetto, contenuto,...

```
public function sendEmail($emailNewUser, $newPassword, $name, $surname){
    $this->mail->setFrom('gestioneacquarimarin@gmail.com', 'Mattia Ruberto');
    $this->mail->addAddress($emailNewUser, $name." ".$surname);
    $this->mail->Subject = 'Gestione Acquari Marini';
    $this->mail->AltBody = "<b>Benvenuto ".$name." ".$surname."</b>";
    $this->mail->Body = "Il tuo account è stato creato con successo, adesso non
dovrai fare altro che accedere al sito con la password di default: <br>
    . $newPassword .
    "<br> e cambiare la password con una tua personale,<br> Cordiali saluti,<br>
Mattia</p>";
    if (!$this->mail->send()) {
        echo 'Mailer Error: ' . $this->mail->ErrorInfo;
    } else {
        echo 'Message sent!';
    }
}
```

Ci ho messo un po' a capire come funzionava perché all'inizio non facevo lo use delle classi della PHPMailer, poi ho avuto altri problemi ma ho risolto.

```
use PHPMailer\PHPMailer\PHPMailer;  
use PHPMailer\PHPMailer\SMTP;  
use PHPMailer\PHPMailer\Exception;
```

Problemi riscontrati e soluzioni adottate

Non facevo lo use delle classi PHPMailer.

Punto della situazione rispetto alla pianificazione

In anticipo di 12 ore.

Programma di massima per la prossima giornata di lavoro

-Sistemare la pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	05.11.2019

## Lavori svolti

Oggi ho ripreso a lavorare da dove mi ero lasciato settimana scorsa, prima ho dovuto fare l'aggiornamento di PHPStorm, poi ho iniziato testando l'invio dell'email. All'inizio non riuscivo a fare il login poi mi sono accorto che non criptavo la password prima di inserirla nel database quindi quando andavo a verificarla non funzionava. Testando con un po' di email differenti nella creazione dell'email mi sono accorto che **webmail1.infomaniak.ch** non supporta come gli invio l'email, infatti esce fuori l'html e non mostra la password ma è un problema della piattaforma quindi se mi avanza tempo ci guardo dietro ma non penso ci sia qualcosa da fare.

**Oggetto:** Gestione Acquari Marini

**Da:** Mattia Ruberto <gestioneacquarimarini@gmail.com> 

**A:** Mattia Ruberto <mattia.ruberto@samtrevano.ch> 

**Data:** 05/11/2019 14:25:10

<b>Benvenuto Mattia Ruberto</b>

Mentre con gmail funziona alla perfezione:

**Mattia Ruberto** <gestioneacquarimarini@gmail.com>

13:57

a me ▾

Il tuo account è stato creato con successo, adesso non dovrà fare altro che accedere al sito con la password di default: P}{JLGy?  
e cambiare la password con una tua personale,  
Cordiali saluti,  
Mattia

Quindi l'aggiunta dell'utente funziona quindi sono passato alla convalidazione dei dati e riguardando ciò che avevo fatto mi è venuta in mente un metodo più pulito per fare la convalidazione, quindi lato client farò i controlli istantaneamente con un onchange su ogni input che richiama i metodi di convalidazione con javascript, mentre quando viene schiacciato il bottone partirà il controllo lato server.

```
function validateEmail(){
    var email = document.getElementById("email").value;
    var reg = /^[(([^>()[]\.,;:\s@"]+([^\s@"][(\.\[^>()[]\.,;:\s@"]+)*])|(\".+\")|((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-\-0-9]+\.)+[a-zA-Z]{2,}))$/;
    if (reg.test(email) || email.length > 0) {
        document.getElementById("email").style.border = "1px solid green";
    } else {
        document.getElementById("email").style.border = "1px solid red";
    }
}

function validateTankName(){
```

```

var tankName = document.getElementById("tankName").value;
var reg = /^[0-9a-zA-Z]+$/i;
if (reg.test(tankName) && tankName.length > 0 && tankName.length <= 45 ) {
    document.getElementById("tankName").style.border = "1px solid green";
} else {
    document.getElementById("tankName").style.border = "1px solid red";
}

function validatePhone(){
    var phoneNumber = document.getElementById("phoneNumber").value;
    var reg = /^[0-9\s\+\#]+$/i;
    if(phoneNumber.length >= 9 && phoneNumber.length <= 30 && reg.test(phoneNumber)){
        document.getElementById("phoneNumber").style.border = "1px solid green";
    }else{
        document.getElementById("phoneNumber").style.border = "1px solid red";
    }
}

function validateDate(){
    var date = document.getElementById("waterChange").value;
    var validateDate = false;
    if (date.length == 10) {
        year = date.substring(0, 4);
        month = date.substring(5, 7) - 1; // because months in JS start from 0
        day = date.substring(8, 10);

        var today = new Date();
        var dayNow = today.getDate();
        var monthNow = today.getMonth() + 1; //January is 0!
        var yearNow = today.getFullYear();

        if(year == yearNow){
            if(monthNow == month){
                if(dayNow < day){
                    validateDate = true
                }
            }else{
                if(day > 0 && day < 31){
                    validateDate = true;
                }
            }
        }else if(year < yearNow){
            if(month > 0 && month < 13){
                if(day > 0 && day < 31){
                    validateDate = true;
                }
            }
        }
    }
    if(validateDate){
        document.getElementById("waterChange").style.border = "1px solid green";
    }else{
        document.getElementById("waterChange").style.border = "1px solid red";
    }
}

function validateKh(){
    var kh = document.getElementById("kh").value;
    if(!validateNumber(kh)){

```

```
document.getElementById("kh").style.border = "1px solid red";
confirmValidate = false;
} else{
    document.getElementById("kh").style.border = "1px solid green";
}
}

function validateLiter(){
    var liter = document.getElementById("liter").value;
    if(!validateNumber(liter)){
        document.getElementById("liter").style.border = "1px solid red";
    } else{
        document.getElementById("liter").style.border = "1px solid green";
    }
}

function validateMagnesium(){
    var magnesium = document.getElementById("magnesium").value;
    if(!validateNumber(magnesium)){
        document.getElementById("magnesium").style.border = "1px solid red";
    } else{
        document.getElementById("magnesium").style.border = "1px solid green";
    }
}

function validateCalcium(){
    var calcium = document.getElementById("calcium").value;
    if(!validateNumber(calcium)){
        document.getElementById("calcium").style.border = "1px solid red";
    } else{
        document.getElementById("calcium").style.border = "1px solid green";
    }
}
```

#### Problemi riscontrati e soluzioni adottate

Dopo aver inviato l'email ho messo un header per tornare alla pagina gestione utenti, ma mi usciva fuori questo errore.

Message sent!

**Warning:** Cannot modify header information - headers already sent by (output started at C:\xampp\htdocs\scuola\ProgettoGestioneAcquariMarini\GestioneAcquariMarini\libs\PHPMailer\SMTP.php:266) in C:\xampp\htdocs\scuola\ProgettoGestioneAcquariMarini\GestioneAcquariMarini\controllers\userManagement.php on line 53

Per risolvere questo sono andato nella riga 266 del file SMTP.php e ho commentato tutto in questo modo non c'è nessun conflitto.

```
//case 'html':  
    //Cleans up output a bit for a better looking, HTML-safe output  
    /*echo gmdate('Y-m-d H:i:s'), ' ', htmlentities(  
        preg_replace('/[\r\n]+/', '', $str),  
        ENT_QUOTES,  
        'UTF-8'  
    ), "<br>\n";*/  
    //break;
```

Punto della situazione rispetto alla pianificazione

In anticipo di 12 ore.

Programma di massima per la prossima giornata di lavoro

-Sistemare la pagina gestione utenti.

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	07.11.2019

## Lavori svolti

Oggi ho finito la convalidazione lato server e client della aggiunta e modifica delle vasche. Infatti ho modificato due cosette come l'aggiunta di un minimo e un massimo di caratteri per le stringhe e per i numeri, li ho ritestati poi ho fatto il backend.

Ho creato 2 classi per farlo:

ValidationFunction => in cui metto le funzioni generali

TankValidation => in cui faccio la validazione della vasca che sta per essere aggiunta o modificata

TankValidation:

```
class TankValidation
{
    private $validationFunction;
    public $stringErrors;

    public function __construct()
    {
        require_once "validationFunction.php";
        $this->validationFunction = new ValidationFunction();
    }

    public function validation($tank){
        $validationOK = true;
        $this->stringErrors = "";

        if(!$this->validationFunction->validateTankName($tank["tankName"])){
            $this->stringErrors = "Il nome dell'acquario è sbagliato";
            $validationOK = false;
        }
        if(!$this->validationFunction->validateInt($tank["magnesium"], 0, 3000)){
            $this->stringErrors .= "<br>Il valore del magnesio è sbagliato";
            $validationOK = false;
        }
        if(!$this->validationFunction->validateInt($tank["calcium"], 0, 1000)){
            $this->stringErrors .= "<br>Il valore del calcio è sbagliato";
            $validationOK = false;
        }
        if(!$this->validationFunction->validateInt($tank["kh"], 0, 20)){
            $this->stringErrors .= "<br>Il valore del kh è sbagliato";
            $validationOK = false;
        }
        if(!$this->validationFunction->validateDate($tank["waterChange"])){
            $this->stringErrors .= "<br>La data è sbagliata";
            $validationOK = false;
        }
        if(!$this->validationFunction->validateInt($tank["liter"], 0, 1000000)){
            $this->stringErrors .= "<br>Il litraggio della vasca è sbagliato";
            $validationOK = false;
        }
    }

    return $validationOK;
}
```

```
}
```

Ho modificato anche il controller tankManagement:

Questo metodo fa il require del form per aggiungere le vasche.

```
public function requirePageAddForm(){
    $title = "Aggiungi vasca";
    $nameButton = "Aggiungi";
    $path = URL . "tankManagement/addTank";
    $stringErrors = $this->tankValidationModel->stringErrors;

    if($this->arrayTank!=null){
        $tankName = $this->arrayTank["tankName"];
        $calcium = $this->arrayTank["calcium"];
        $magnesium = $this->arrayTank["magnesium"];
        $kh = $this->arrayTank["kh"];
        $waterChange = $this->arrayTank["waterChange"];
        $liter = $this->arrayTank["liter"];
    }

    require "GestioneAcquariMarini/views/_templates/header.php";
    require "GestioneAcquariMarini/views/_templates/menu.php";
    require "GestioneAcquariMarini/views/gestioneAcquari/tank/tankManagement.php";
    require "GestioneAcquariMarini/views/_templates/footer.php";
}
```

Questo è il metodo richiamato quando l'utente preme il bottone aggiungi

```
public function formAddTank(){
    session_start();
    if($_SESSION["authentification"] == true){
        $this->requirePageAddForm();
    }else{
        header("Location:".URL);
    }
}
```

Metodo che mi ritorna i valori negli input:

```
private function getTankArray(){
    $name = $_POST["tankName"];
    $magnesium = $_POST["magnesium"];
    $calcium = $_POST["calcium"];
    $kh = $_POST["kh"];
    $waterChange = $_POST["waterChange"];
    $liter = $_POST["liter"];

    $tank = array("tankName"=>$name, "magnesium"=>$magnesium, "calcium"=>$calcium,
    "kh"=>$kh, "waterChange"=>$waterChange, "liter"=>$liter);
    return $tank;
}
```

Nel tankModel ho aggiunto una funzione che mi ritorna tutti i nomi delle vasche presenti sul database per evitare che venga inserita un nome già esistente:

```
public function getAllTankName(){
    $selectThank = "select nome from vasca";
    $allName = $this->executeAndFetchStatement($selectThank);
    return $allName;
}
```

Nella classe validation ho aggiunto e modificato in particolare queste funzioni:

```
public function validateTankName($tankName)
{
    require_once "tankModel";
    $tankModel = new TankModel();
    $validElement = $this->generalValidation($tankName);
    $pattern = '/^([A-Za-z0-9_-])*$/';
    if (preg_match($pattern, $validElement) && strlen($validElement) > 0 &&
    strlen($validElement) <= 45) {
        $allName = $tankModel->getAllTankName();
        if (!in_array($tankName, $allName)){
            return true;
        }else{
            return false;
        }
    }else{
        return false;
    }
}

public function validateDate($date){
    $date_arr = explode('-', $date);
    $current_date = new DateTime();
    $dateOk = false;
    if (count($date_arr) == 3) {
        if (checkdate($date_arr[2], $date_arr[1], $date_arr[0])) {
            if ($date > $current_date) {
                $dateOk = true;
            }
        }
    }
    return $dateOk;
}
```

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In anticipo di una settimana

#### Programma di massima per la prossima giornata di lavoro

-Fare la convalidazione lato client e server per gli utenti

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	08.11.2019

## Lavori svolti

Ho modificato la validazione del nome della vasca:

```
public function validateTankName($tankName)
{
    $validElement = $this->generalValidation($tankName);
    $pattern = '/^([A-Za-z0-9_-])*$/';
    $arrayAllNameTank = $this->multidimensionalArrayToNormalArray($this-
>allNameTank);

    if (preg_match($pattern, $validElement) && strlen($validElement) > 0 &&
    strlen($validElement) <= 45) {
        if (in_array($tankName, $arrayAllNameTank)){
            return self::ALREADY_EXIST;
        }else{
            return self::CORRECT_NAME;
        }
    }else{
        return self::INCORRECT_NAME;
    }
}
```

Nel tankValidation:

```
if($this->validationFunction->validateTankName($tank["tankName"]) ==
ValidationFunction::ALREADY_EXIST){
    $this->stringErrors = "Il nome della vasca esiste già";
    $validationOK = false;
}else if($this->validationFunction->validateTankName($tank["tankName"]) ==
ValidationFunction::INCORRECT_NAME){
    $this->stringErrors = "Il nome della vasca è sbagliato";
    $validationOK = false;
}
```

Ho modificato anche le funzioni della modifica per far sì che la validazione funzioni anche quando si modificano i dati degli acquari.

```
public function requirePageModifyForm($name){
    $title = "Modifica vasca";
    $nameButton = "Modifica";
    $path = URL . "tankManagement/modifyTank/".$name;
    $stringErrors = $this->tankValidatioModel->stringErrors;

    if($this->arrayTank!=null){
        $tankName = $this->arrayTank["tankName"];
        $calcium = $this->arrayTank["calcium"];
        $magnesium = $this->arrayTank["magnesium"];
        $kh = $this->arrayTank["kh"];
        $waterChange = $this->arrayTank["waterChange"];
        $liter = $this->arrayTank["liter"];
    }else{
        $tankToModify = $this->tankManagementModel->getByName($name);
        $tankName = $tankToModify[0]["nome"];
    }
}
```

```
$magnesium = $tankToModify[0]["calcio"];
$calcium = $tankToModify[0]["magnesio"];
$kh = $tankToModify[0]["kh"];
$waterChange = $tankToModify[0]["ultimo_cambio_acqua"];
$liter = $tankToModify[0]["Litri"];
}

require "GestioneAcquariMarini/views/_templates/header.php";
require "GestioneAcquariMarini/views/_templates/menu.php";
require "GestioneAcquariMarini/views/gestioneAcquari/tank/tankManagement.php";
require "GestioneAcquariMarini/views/_templates/footer.php";
}
public function formModifyTank($name)
{
    session_start();
    if ($_SESSION["authentification"] == true) {
        $this->requirePageModifyForm($name);
    } else {
        header("Location: " . URL);
    }
}
```

Poi ho iniziato a fare la stessa cosa per la convalidazione dell'utente nello stesso modo.

#### Problemi riscontrati e soluzioni adottate

Oggi mi sono bloccato mentre stavo Sistema la validazione del nome della vasca infatti nel controllare se il nome ce già il nome usavo la seguente funzione:

```
if (in_array($tankName, $arrayAllNameTank)){
    return self::ALREADY_EXIST;
} else{
    return self::CORRECT_NAME;
}
```

E la funzione `in_array` non funzionava e non capivo il motivo, così ho perso un sacco di tempo a capire il motivo per cui non funzionava utilizzando anche `array_search` e testando il più possibile tutti i valori che inserivo, alla fine ho capito che tutte le funzioni di php che ti permettono di capire se un valore è già presente nell'array non accettano array multidimensionali, per risolvere il problema ho fatto questa funzione che converte l'array multidimensionale in un normale array.

```
private function multidimensionalArrayToNormalArray($arrayMultidimensional){
    $result = array();
    foreach ($arrayMultidimensional as $key => $value) {
        $result[] = $value["nome"];
    }
    return $result;
}
```

#### Punto della situazione rispetto alla pianificazione

In anticipo di una settimana

#### Programma di massima per la prossima giornata di lavoro

-Finire la convalidazione lato client e server per gli utenti

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	08.11.2019

## Lavori svolti

Ho fatto la convalidazione lato client dell'utente, ho fatto le funzioni di convalidazione lato server:

```
public function validatePhoneNumber($phoneNumber)
{
    $validElement = $this->generalValidation($phoneNumber);
    $pattern = '/^[\a-zéëäääööüü\s]*$/i';

    if (strlen($phoneNumber) > 0 && strlen($phoneNumber) <= 45 &&
preg_match($pattern, $validElement)) {
        return true;
    } else {
        return false;
    }
}

public function validateString($string){
    $validElement = $this->generalValidation($string);
    $pattern = '/^[\d\s\#]+$/i';

    if (strlen($validElement) > 0 && strlen($validElement) <= 45 &&
preg_match($pattern, $validElement)) {
        return true;
    } else {
        return false;
    }
}

public function validatePermission($permission){
    if($permission == "User" || $permission == "Admin"){
        return true;
    }else{
        return false;
    }
}

public function validateEmail($email){
    $validElement = $this->generalValidation($email);
    $pattern =
'^/(([^<>()\\\[\\,.\\:\\s@\\"]+\\(.[^<>()\\\[\\,.\\:\\s@\\"]+)*|\\(.+\\")\\@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\])|(([a-zA-Z\\-0-9]+\\.)+[a-zA-Z]{2,}))$/';

    if( preg_match($pattern, $validElement)){
        return true;
    }else{
        return false;
    }
}
```

Poi ho iniziato ad implementare la validazione dell'utente con le varie funzioni e nel mentre mi sono venute un po' di idee per ottimizzare il codice

quindi mi sono messo a fare quello.

Ho ottimizzato il metodo per fare il require della pagina in questo modo ho solo una funzione invece di averne due che facevano la stessa cosa:

```
public function requirePageForm($pageInformation){
    $title = $pageInformation[0];
    $nameButton = $pageInformation[1];
    $path = $pageInformation[2];
    $stringErrors = $pageInformation[3];
    $name = null;
    if(count($pageInformation) > 4){
        $name = $pageInformation[4];
    }
    if($this->arrayTank!=null){
        $tankName = $this->arrayTank["tankName"];
        $calcium = $this->arrayTank["calcium"];
        $magnesium = $this->arrayTank["magnesium"];
        $kh = $this->arrayTank["kh"];
        $waterChange = $this->arrayTank["waterChange"];
        $liter = $this->arrayTank["liter"];
    }else if($name != null){
        $tankToModify = $this->tankManagementModel->getByName($name);
        $tankName = $tankToModify[0]["nome"];
        $magnesium = $tankToModify[0]["calcio"];
        $calcium = $tankToModify[0]["magnesio"];
        $kh = $tankToModify[0]["kh"];
        $waterChange = $tankToModify[0]["ultimo_cambio_acqua"];
        $liter = $tankToModify[0]["Litri"];
    }
    require "GestioneAcquariMarini/views/_templates/header.php";
    require "GestioneAcquariMarini/views/_templates/menu.php";
    require "GestioneAcquariMarini/views/gestioneAcquari/tank/tankManagement.php";
    require "GestioneAcquariMarini/views/_templates/footer.php";
}
```

### Problemi riscontrati e soluzioni adottate

Array ( [tankName] => zzzz [magnesium] => 0 [calcium] => 0 [kh] => 20 [waterChange] => 2000-11-11 [liter] => 33 ) vasca1

Fatal error: Uncaught PDOException: SQLSTATE[23000]: Integrity constraint violation: 1062 Duplicate entry " for key 'PRIMARY'

Mi sono accorto che quando modifico la vasca e modifco anche il nome cioè la primary key mi dice che ce un errore e che la primary key è duplicata, solo che non capisco qual'è il problema dato che il codice che ho scritto è giusto dato che facendo la stessa cosa tramite interfacci grafica da MySQL Workbench il codice utilizzato è lo stesso quindi non capisco qual'è il problema.

### Punto della situazione rispetto alla pianificazione

In anticipo di una settimana

### Programma di massima per la prossima giornata di lavoro

-Finire la convalidazione lato server per gli utenti

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	14.11.2019

## Lavori svolti

Ho sistemato la convalidazione della data e dell'email, che davano entrambi problemi:

```
public function validateDate($stringDate){
    $date_arr = explode('-', $stringDate);
    $date = new DateTime($stringDate);
    $current_date = new DateTime();
    $dateOk = false;
    if (count($date_arr) == 3 && strlen($stringDate) == 10) {
        if (checkdate($date_arr[2], $date_arr[1], $date_arr[0])) {
            if ($date <= $current_date) {
                $dateOk = true;
            }
        }
    }
    return $dateOk;
}

public function validateEmail($email){
    $validElement = $this->generalValidation($email);
    $pattern =
'/^(([^<>()[]\\.,;:\\s@\\"]+([^.^<>()[]\\.,;:\\s@\\"]+)*|(.+\\")|((\\[[0-9]{1,3}\\.[0-
9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\])|(([a-zA-Z\\-0-9]+\\.)+[a-zA-Z]{2,}))$/';
    $arrayAllEmailUser = $this->multidimensionalArrayToNormalArray($this-
>allEmailUsers);
    if( preg_match($pattern, $validElement)){
        if (in_array(strtolower($email), $arrayAllEmailUser)){
            return self::ALREADY_EXIST;
        }else{
            return self::CORRECT_INPUT;
        }
    }else{
        return self::INCORRECT_INPUT;
    }
}
```

Ho fatto delle costanti per gestire dei valori, abitudine che devo prendere maggiormente per avere una coerenza dei dati.

Poi ho continuato l'implementazione della convalidazione, ho quasi finito mi mancano due cose da fare.

## Problemi riscontrati e soluzioni adottate

Ho risolto il problema dell'update, il problema era nella convalidazione infatti io quando andavo a guardare se il nome era già presente distinguevo le lettere maiuscole e minuscole mentre mysql non fa questa distinzione quindi mi dava errore e io non lo vedevo.

Punto della situazione rispetto alla pianificazione

In anticipo di una settimana

Programma di massima per la prossima giornata di lavoro

Finire la convalidazione lato server per gli utenti

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	15.11.2019

## Lavori svolti

Oggi ho completato tutto ciò che riguarda la gestione delle vasche e degli utenti a meno di eventuali imprevisti. Ho aggiornato il forum dell'utente dato che avevo fatto copia e incolla da quello degli acquari e alcuni id e name non li avevo cambiati, avevo invertito i regex della validazione email e delle stringhe quindi ho sistemato anche questo, per selezionare le opzioni nelle select ho usato questo codice:

```
<div class="form-group">
    <label for="formGroupExampleInput">Cambio Password</label>
    <select class="form-control" id="passwordChange" name="passwordChange"
onchange="validateSelectChangePassword(this)">
        <option></option>
        <option ><?php if(isset($userPasswordChange) && $userPasswordChange == 0){echo
'selected';} ?> <?php echo TOCHANGEPASSWORD ?></option>
        <option ><?php if(isset($userPasswordChange) && $userPasswordChange == 1){echo
'selected';} ?> <?php echo NOTCHANGEPASSWORD ?></option>
    </select>
</div>
```

Ho aggiunto una funzione in userModel:

```
public function getByName($name){
    $selectTankByName = "Select * FROM vasca WHERE Nome=:nameTank";
    $this->statement = $this->connection->prepare($selectTankByName);
    $this->statement->bindParam(':nameTank', $name , PDO::PARAM_STR);
    $this->statement->execute();
    $result = $this->statement->fetchAll(PDO::FETCH_ASSOC);
    return $result;
}
```

Poi ho corretto gli errori e testato l'applicazione il più possibile, ho aggiunto un controllo per determinare se l'email esiste o no:

```
if($this->confirmChangementEmail($this->arrayUser['email'])){
    $this->usersManagementModel->add($this->arrayUser);
    $this->arrayUser = null;
    header("Location:" . URL . "userManagement");
} else{
    $path = URL . "userManagement/modifyUser/".$email;
    $stringErrors = "L'email non esiste<br>".$this->userValidationModel-
>stringErrors;
    $pageInformation = array("Modifica utente", "Modifica", $path, $stringErrors,
$email);
    $this->requirePageForm($pageInformation);
}
```

Ho aggiunto una nuova funzione che viene usata quando viene modificata l'email di un utente, questo è per confermare il cambio di email e che l'email sia reale.

```
public function sendEmailModifyUser($emailUser){
    $this->mail->setFrom('gestioneacquarimarini@gmail.com', 'Mattia Ruberto');
    $this->mail->addAddress($emailUser);
    $this->mail->Subject = 'Gestione Acquari Marini';
    $this->mail->AltBody = "<b>Conferma email</b>";
```

```
$this->mail->Body = "La sua email è stata aggiornata con successo";
if ($this->mail->send()) {
    return true;
} else {
    return false;
}
```

Nella classe PHPMailer ho commentato tutte le righe che stampano output per evitare che se l'email non sia reale mi escano tutte le informazioni che la classe fa in automaticamente sulla pagina.

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In anticipo di una settimana

#### Programma di massima per la prossima giornata di lavoro

Implementare un modo per gestire se l'utente dimentica la password

Implementare i trigger sul database per far sì che i valori diminuiscono

Inviare un email quando i valori escono dai range

Gestire gli abitanti delle vasche

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	19.11.2019

## Lavori svolti

Oggi ho implementato l'evento nel database che si ripete ogni 7 giorni e fa diminuire i valori delle vasche, il magnesio diminuisce di 100, il kh di 1 e il calcio di 25. Questo è il codice che ho usato per creare l'evento, per ciclare tutte le righe della tabella ho usato un cursore.

```
DROP EVENT IF EXISTS updateValueTank;
```

```
DELIMITER $$$
CREATE EVENT updateValueTank ON SCHEDULE EVERY 7 DAY DO BEGIN
    DECLARE nomeVasca VARCHAR(50);
    DECLARE calcioVasca INT;
    DECLARE magnesioVasca INT;
    DECLARE khVasca INT;

    DECLARE cur CURSOR FOR SELECT nome FROM vasca;

    OPEN cur;
    read_loop: LOOP
        FETCH NEXT FROM cur INTO nomeVasca;
        SELECT magnesio into magnesioVasca FROM vasca WHERE nome=nomeVasca;
        SELECT calcio into calcioVasca FROM vasca WHERE nome=nomeVasca;
        SELECT kh into khVasca FROM vasca WHERE nome=nomeVasca;
        UPDATE vasca SET kh=(khVasca-1),calcio=(calcioVasca-25),magnesio=(magnesioVasca-100) WHERE (nome=nomeVasca);

    END LOOP;
    CLOSE cur;
END $$$
DELIMITER;
```

Poi ho guardato per la gestione dei valori in particolare quando escono da loro range idoneo devo poter inviare un email all'amministratore e discutendo con il mio formatore sulle varie possibilità l'unico modo per poter far partire un'applicazione in background in qualsiasi caso anche se il sito web non viene mai aperto è utilizzando la funzione di windows "Utilità di pianificazione", in questo modo si può pianificare che l'applicazione faccia partire uno script che richiama il metodo di controllo e questo metodo se c'è bisogno invia un email all'amministratore.

## Problemi riscontrati e soluzioni adottate

Non mi ricordavo bene come usare le procedure, trigger e eventi e in particolare i cursori, quindi mi sono dovuto ripetere un po' tutto per riuscire a implementare l'evento che ogni 7 giorni fa diminuire di un tot i valori delle vasche.

## Punto della situazione rispetto alla pianificazione

In anticipo di una settimana.

Programma di massima per la prossima giornata di lavoro

- Implementare un modo per gestire se l'utente dimentica la password
- Inviare un email quando i valori escono dai range
- Gestire gli abitanti delle vasche

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	21.11.2019

## Lavori svolti

Ho fatto la classe per il controllo dei valori:

```
<?php

class tankValueControl {
    private $mailModel;
    private $acquariums;
    private $sendEmail = false;
    private $password = '$2y$10$AEG07IpNe01KNOPuShvPGOf2M1fNUHX4aHU1RVhr1/5bBx4/a.FwS';

    public function __construct(){
        require_once "GestioneAcquariMarini/models/tankModel.php";
        require_once "GestioneAcquariMarini/models/mailModel.php";
        $tankManagementModel = new TankModel();
        $this->mailModel = new MailModel();
        $this->acquariums = $tankManagementModel->getAll();
    }

    public function valueControl($password){
        if(password_verify($password, password_hash("Password@1@1234",
PASSWORD_DEFAULT))) {
            foreach ($this->acquariums as $tank) {
                $name = $tank["nome"];
                $magnesium = $tank["magnesio"];
                $calcium = $tank["calcio"];
                $kh = $tank["kh"];
                $message = "I valori della vasca " . $name . " non sono nel range
adatto, in particolare:";

                if (!$this->checkRangeValue($magnesium, 1200, 1450)) {
                    $message .= "<br> Megnesio: " . $magnesium;
                    $this->sendEmail = true;
                }
                if (!$this->checkRangeValue($calcium, 350, 450)) {
                    $message .= "<br> Calcio: " . $calcium;
                    $this->sendEmail = true;
                }
                if (!$this->checkRangeValue($kh, 7, 11)) {
                    $message .= "<br> Kh: " . $kh;
                    $this->sendEmail = true;
                }
                if ($this->sendEmail) {
                    $this->mailModel->sendEmailWarning($name, $message);
                    echo "Email inviata per la vasca: ".$name;
                    echo "<br>";
                }else{
                    echo "Nessuna email inviata per la vasca: ".$name;
                }
            }
        }else{
    }
}
```

```
        echo "Password sbagliata";
    }
    echo "<script type=\"text/javascript\">close_window()</script>";
}

public function checkRangeValue($calcium, $min, $max){
    if($calcium >= $min && $calcium <= $max){
        return true;
    }else{
        return false;
    }
}
?>
```

Ho fatto lo script per fare partire la pagina web contenente la seguente riga:

start

<http://localhost/scuola/ProgettoGestioneAcquariMarini/tankValueControl/valueControl/PASSWORD@1@1234;>

Ho creato l'attività di pianificazione.

Poi volevo che una volta finita l'operazione chiude la pagina in automatico m non ho avuto tempo.

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In anticipo di 18ore.

#### Programma di massima per la prossima giornata di lavoro

-Implementare un modo per gestire se l'utente dimentica la password  
-Gestire gli abitanti delle vasche

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	22.11.2019

## Lavori svolti

Oggi ho aggiunto questa riga alla fine del metodo del controllo dei valori in questo modo dopo aver fatto il controllo e se è il caso inviato l'email la pagina nel browser si chiude e non rimangono aperte ad ogni controllo che fa.

```
echo '<script type="text/javascript">
    var win = window.open("about:blank", "_self");
    win.close();
</script>';
```

Ho aggiunto i controlli per gli utenti user che non devono poter accedere alla pagina gestione utenti e non possono aggiungere vasche.

```
if($_SESSION["type"] == "Admin") {
    $stringErrors = $this->tankValidatioModel->stringErrors;
    $path = URL . "tankManagement/addTank";
    $pageInformation = array("Aggiungi vasca", "Aggiungi", $path, $stringErrors);
    $this->requirePageForm($pageInformation);
} else{
    header("Location:" . URL . "home");
}
```

con un semplice controllo della sessione istanziata al login con al suo interno il tipo di utente. Questi controlli li ho messi in tutti i posti dove ci sono require, dovrò aggiungere questo controllo anche per tutti gli altri require infatti devo controllare che l'utente sia autenticato sempre.

Poi ho iniziato a fare l'implementazione per gestire se un utente dimentica la password.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In anticipo di 18ore.

## Programma di massima per la prossima giornata di lavoro

-Implementare un modo per gestire se l'utente dimentica la password  
-Gestire gli abitanti delle vasche

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	26.11.2019

## Lavori svolti

Oggi ho gestito il fatto che l'utente può dimenticare l'email, quindi sotto gli input del login ce un bottone "ho dimenticato la password" e quando viene cliccato compare un prompt che richiede l'email a cui si vuole cambiare la password, viene eseguito il controllo dell'email e se è corretta e già presente nel database viene generata una nuova password di default che poi viene impostata nel database e viene inviata per email all'utente.

Quando viene schiacciato il bottone viene fatta partire una funzione javascript:

```
function changePassword(){
    var emailToChange = prompt("Per favore inserisci la tua email:");
    if(emailToChange.length > 0) {
        window.location.href = URL+"login/updatePassword/"+emailToChange;
    }else{
        alert("Devi inserire per forza un email");
    }
}
```

Che poi porta al controller del login per l'update della password

```
public function updatePassword($email){
    $newPasswordGenerate = $this->userModel->generetaRandomPassword();
    $passwordHash = password_hash($newPasswordGenerate, PASSWORD_DEFAULT);
    if($this->userValidation->validateEmail($email)){
        $this->userModel->updatePassword($email, $passwordHash);
        $this->mailModel->sendEmailUpdatePassword($email, $newPasswordGenerate);
        $_SESSION["errorRequestNewPassword"] = 1;
    }else{
        $_SESSION["errorRequestNewPassword"] = 2;
    }
    header("Location:".$URL."login");
}
```

Nello user model ho aggiunto questo metodo che mi fa un update della password e del flag per il cambio password

```
public function updatePassword($email,$password){
    $changePassword = 0;
    $updatePassword = "UPDATE utente SET cambioPassword=:cambioPassword,
password=:password WHERE (email=:email)";
    $this->statement = $this->connAccess->prepare($updatePassword);
    $this->statement->bindParam(':cambioPassword', $changePassword, PDO::PARAM_STR);
    $this->statement->bindParam(':password', $password , PDO::PARAM_STR);
    $this->statement->bindParam(':email', $email , PDO::PARAM_STR);
    $this->statement->execute();
}
```

```
Poi ho implementato un nuovo metodo nel MailModel  
public function emailUpdatePassword($emailUser, $newPassword){  
    $this->mail->addAddress($emailUser);  
    $this->mail->AltBody = "<b>La tua password è stata aggiornata con successo<b>";  
    $this->mail->Body = "Il tua password è stata aggiornata con successo, adesso non  
dovrai fare altro che accedere al sito con la password di default: <br>"  
        . $newPassword .  
        "<br> e cambiare la password con una tua personale,<br> Cordiali saluti,<br>  
Mattia</p>";  
    return $this->sendEmail();  
}
```

Dato ripeteva sempre la stessa per inviare l'email ho aggiunto questo metodo:

```
public function sendEmail(){  
    if ($this->mail->send()) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

E ho messo alcune variabili per inviare l'email nel costruttore.

Poi ho fatto in modo che l'utente loggato non può autoelimanarsi, ho bloccato i buttoni per utenti normali di aggiunta e rimozione delle vasche.

Ho sistemato altri piccoli particolari ed infine ho iniziato l'implementazione della gestione degli abitanti.

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In anticipo di 4ore.

#### Programma di massima per la prossima giornata di lavoro

-Finire la gestione degli abitanti delle vasche

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	28.11.2019

## Lavori svolti

Oggi ho continuato l'implementazione degli abitanti, nel frattempo mi sono accorto che mi mancava una tabella per sapere a quella vasca appartengono gli abitanti, quindi ho fatto la tabella.

```
use gestioneacquarimarini;
CREATE TABLE contiene (
    nome_vasca VARCHAR(45) NOT NULL,
    specie_abitante VARCHAR(45) NOT NULL,
    genere_abitante VARCHAR(45) NOT NULL,
    PRIMARY KEY (nome_vasca,specie_abitante,genere_abitante),
    FOREIGN KEY (nome_vasca) REFERENCES vasca(nome),
    FOREIGN KEY (specie_abitante, genere_abitante) REFERENCES abitante(specie, genere)
);
```

Ho perso un sacco di tempo perché invece di fare una foreign key singola per i due valori ne facevo due singole e non capivo qual'era il problema, quando ho finito e ho risolto il problema mi è venuto in mente che una terza tabella non serve e che basta aggiungere il nome della vasca all'interno della tabella abitante per capire a quale vasca appartengono gli abitanti quindi ho perso tempo inutilmente. Inoltre sono venuti due volte i ragazzi di quarta media in classe e ho dovuto spiegargli ciò che stavo facendo.

Quindi poi per non perdere altro tempo ho cancellato e ricreato la tabella abitante per non stare li a usare alter table.

```
use gestioneacquarimarini;
CREATE TABLE abitante (
    specie VARCHAR(45) NOT NULL,
    genere VARCHAR(15) NOT NULL,
    tipo VARCHAR(45) NOT NULL,
    nome_vasca VARCHAR(45) NOT NULL,
    numero INT NOT NULL,
    PRIMARY KEY (specie,genere),
    FOREIGN KEY (nome_vasca) REFERENCES vasca(nome)
);
```

Poi ho implementato la tabella, il bottone "form gestione abitante" mostra o nasconde il form per aggiungere e modificare gli abitanti della vasca.

Gestione Acquari Marini    Riassuntiva    Gestione vasche    Gestione utenti    Logout

## Pagina Gestione Abitanti

Nome Vasca: Vasca66

**Form gestione abitante**

Specie	<input type="text"/>	Sesso	<input type="text"/>	Tipo	<input type="text"/>	Numero	<input type="text"/>	<b>Aggiungi</b>
--------	----------------------	-------	----------------------	------	----------------------	--------	----------------------	-----------------

Nome Specie	Sesso	Tipo	Numero	Modifica	Rimuovi
Pesce pagliaccio	M	Pesce	30	<b>Modifica</b>	<b>Rimuovi</b>

© copyright Mattia Ruberto

### Problemi riscontrati e soluzioni adottate

-Non potevo fare due foreign key della stessa tabella separate, risolto facendone una doppia.

### Punto della situazione rispetto alla pianificazione

In ritardo dove finire oggi l'implementazione.

### Programma di massima per la prossima giornata di lavoro

-Finire la gestione degli abitanti delle vasche

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	29.11.2019

## Lavori svolti

Oggi ho continuato l'implementazione degli abitanti, ho modifato alcune funzioni e parametri del habitantModel, questo è il metodo che uso per riempire la tabella iniziale:

```
public function getAllHabitantForTank($tankName)
{
    $selectHabitants = "select * from abitante where nome_vasca=:tankName";
    $this->statement = $this->connection->prepare($selectHabitants);
    $this->statement->bindParam(':tankName', $tankName, PDO::PARAM_STR);
    $this->statement->execute();
    $result = $this->statement->fetchAll(PDO::FETCH_ASSOC);
    return $result;
}
```

Questa funzione la uso per ricavare se esiste già un abitante della stessa specie e dello stesso sesso per la convalidazione e la modifica degli abitanti.

```
public function getAllHabitantBySpeciesAndSex($species, $sex)
{
    $selectHabitants = "Select * FROM abitante WHERE specie=:species AND
genere=:sex";
    $this->statement = $this->connection->prepare($selectHabitants);
    $this->statement->bindParam(':species', $species, PDO::PARAM_STR);
    $this->statement->bindParam(':sex', $sex, PDO::PARAM_STR);
    $this->statement->execute();
    $result = $this->statement->fetchAll(PDO::FETCH_ASSOC);
    return $result;
}
```

Ho implementato il metodo per rimuovere un abitante:

```
public function delete($species, $sex)
{
    $selectHabitants = "DELETE FROM abitante WHERE specie=:species AND genere=:sex";
    $this->statement = $this->connection->prepare($selectHabitants);
    $this->statement->bindParam(':species', $species, PDO::PARAM_STR);
    $this->statement->bindParam(':sex', $sex, PDO::PARAM_STR);
    $this->statement->execute();
}
```

Ho modificato anche l'index e ho aggiunto una nuova funzione che uso di default:

Infatti se per caso l'URL punta all'index il sito ti riporta al login perché non ci puoi arrivare dal sito.

```
public function index(){
    header("Location: ".URL);
}
```

Questo invece è il metodo che mostra la pagina della gestione degli abitanti per ogni vasca:

```
function showAllHabitantsTank($referenceTankName = 1){
    $path = URL . "habitantManagement/addHabitant";
    if($_SESSION["authentification"] == true && strlen($referenceTankName) > 0 &&
$referenceTankName != 1){
        if($this->tankValidation->validateTankName($referenceTankName)) {
            $_SESSION["referencesTankName"] = $referenceTankName;
            $habitants = $this->habitantModel-
>getAllHabitantForTank($_SESSION["referencesTankName"]);
            $nameButton = "Aggiungi";
            require "GestioneAcquariMarini/views/_templates/header.php";
            require "GestioneAcquariMarini/views/_templates/menu.php";
            require
"GestioneAcquariMarini/views/gestioneAcquari/habitants/index.php";
            require "GestioneAcquariMarini/views/_templates/footer.php";
        }else{
            header("Location:".URL);
        }
    }else{
        header("Location:".URL);
    }
}
```

Poi ho fatto la convalidazione nell'aggiunta e l'implementazione, queste sono le funzioni:

```
public function validateSex($sex){
    if ($sex == self::MALE_SEX || $sex == self::FEMALE_SEX || $sex ==
self::OTHER_SEX) {
        return true;
    } else {
        return false;
    }
}

public function validatePrimaryKeysHabitant($species, $sex){
    $this->allHabitants = $this->habitantModel->getAllHabitantBySpeciesAndSex();
    if(count($this->allHabitants) > 0){
        return true;
    }
    return false;
}
```

Poi come per gli altri ho implementato la convalidazione dove uso queste funzioni per la convalidazione della specie uso la convalidazione del nome della vasca che è la stessa cosa.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In ritardo di 4 ore.

Programma di massima per la prossima giornata di lavoro

-Finire la gestione degli abitanti delle vasche

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	03.12.2019

## Lavori svolti

Oggi ho finito l'implementazione della gestione degli abitanti, nel model ho fatto i metodi per aggiungere e modificare i dati nel database:

```
public function add($habitante)
{
    $addHabitante = "INSERT INTO abitante (specie, genere, tipo, numero, nome_vasca) VALUES
(?, ?, ?, ?, ?)";
    $this->statement = $this->connection->prepare($addHabitante);
    $this->statement->bindParam(1, $habitante["species"]);
    $this->statement->bindParam(2, $habitante["sex"]);
    $this->statement->bindParam(3, $habitante["type"]);
    $this->statement->bindParam(4, $habitante["habitanteNumber"]);
    $this->statement->bindParam(5, $_SESSION["referencesTankName"]);
    $this->statement->execute();
}
public function modify($habitante, $species, $sex)
{
    $modifyHabitante = "UPDATE abitante SET specie=:species,
genere=:sex, tipo=:type, numero=:habitanteNumber WHERE specie=:originalSpecies &&
genere=:originalSex && nome_vasca=:tankName";
    $this->statement = $this->connection->prepare($modifyHabitante);
    $this->statement->bindParam(':originalSpecies', $species, PDO::PARAM_STR);
    $this->statement->bindParam(':originalSex', $sex, PDO::PARAM_STR);
    $this->statement->bindParam(':species', $habitante['species'], PDO::PARAM_STR);
    $this->statement->bindParam(':sex', $habitante['sex'], PDO::PARAM_STR);
    $this->statement->bindParam(':type', $habitante['type'], PDO::PARAM_STR);
    $this->statement->bindParam(':habitanteNumber', $habitante['habitanteNumber'],
PDO::PARAM_STR);
    $this->statement->bindParam(':tankName', $_SESSION["referencesTankName"],
PDO::PARAM_STR);
    $this->statement->execute();
}
```

Poi nel controller ho gestito i vari buttoni e le varie azioni che bisogna eseguire, per l'aggiunta ho usato semplicemente questo metodo:

```
public function addHabitante(){
    $this->habitanteManagement = $this->getArrayHabitanteByPost();
    $habitante = $this->habitanteManagement;
    if($this->habitanteValidation->validation($habitante, null, null)){
        $this->habitanteModel->add($habitante);
        $this->habitanteManagement = null;
    }
    $this->stringErrors = $this->habitanteValidation->stringErrors;
    $this->showAllHabitantsTank($_SESSION["referencesTankName"]);
}
```

Mentre per la modifica una funzione che mi riporta i dati da modificare nel form e un che mi faccia la modifica:

```
public function modifyHabitant($species, $sex){
    $habitants = $this->habitantModel->getAllHabitantBySpeciesAndSex($species, $sex);
    $this->habitantArrayManagement = $this->getArrayHabitantByDatabase($habitants);
    $this->textButtonForm = "Modifica";
    $this->pathForm = URL . "habitantManagement/updateHabitant/" . $species . "/" . $sex;
    $this->showAllHabitantsTank($_SESSION["referencesTankName"]);
}

public function updateHabitant($species, $sex){
    $this->habitantArrayManagement = $this->getArrayHabitantByPost();
    $habitants = $this->habitantArrayManagement;
    $habitantsByDatabase = $this->getArrayHabitantByDatabase($this->habitantModel-
        >getAllHabitantBySpeciesAndSex($species, $sex));
    if ($this->habitantValidation->validation($habitants, $habitantsByDatabase["species"],
        $habitantsByDatabase["sex"])){
        $this->habitantModel->modify($habitants, $habitantsByDatabase["species"],
        $habitantsByDatabase["sex"]);
        $this->habitantArrayManagement = null;
        $this->textButtonForm = "Aggiungi";
        $this->pathForm = URL . "habitantManagement/addHabitant";
    } else{
        $this->textButtonForm = "Modifica";
        $this->pathForm = URL . "habitantManagement/updateHabitant/" . $species . "/" . $sex;
    }
    $this->stringErrors = $this->habitantValidation->stringErrors;
    $this->showAllHabitantsTank($_SESSION["referencesTankName"]);
}
```

Per cancellare un vasca prima ho dovuto cancellare tutti gli abitanti correlati alla vasca:

```
public function delete($name){
    $deleteHabitantTank = "DELETE FROM abitante WHERE nome_vasca=:nameTank";
    $this->statement = $this->connection->prepare($deleteHabitantTank);
    $this->statement->bindParam(':nameTank', $name, PDO::PARAM_STR);
    $this->statement->execute();
    $deleteTank = "DELETE FROM vasca WHERE Nome=:nameTank";
    $this->statement = $this->connection->prepare($deleteTank);
    $this->statement->bindParam(':nameTank', $name, PDO::PARAM_STR);
    $this->statement->execute();
}
```

Questi due metodi li uso per avere un array dell'abitante, nel primo caso dal database e nel secondo dal post del form:

```
public function getArrayHabitantByPost(){
    $species = $_POST["species"];
    $sex = $_POST["sex"];
    $type = $_POST["type"];
    $habitantsNumber = $_POST["habitantsNumber"];
    $habitants =
array("species"=>$species, "sex"=>$sex, "type"=>$type, "habitantsNumber"=>$habitantsNumber);
    return $habitants;
}
```

```
public function getArrayHabitantByDatabase($habitant){  
    $species = $habitant[0]["specie"];  
    $sex = $habitant[0]["genere"];  
    $type = $habitant[0]["tipo"];  
    $habitantNumber = $habitant[0]["numero"];  
    $habitant =  
array("species"=>$species, "sex"=>$sex, "type"=>$type, "habitantNumber"=>$habitantNumber);  
    return $habitant;  
}
```

Poi ho fatto alcuni test personalmente e ho fatto fare anche un test del prodotto a Carlo.

Ho sistemato alcune cose come il fatto che per gli utenti normali non esce neanche nel menu la possibilità di andare in gestione utenti.

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

#### Programma di massima per la prossima giornata di lavoro

-Documentare il tut

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	05.12.2019

Lavori svolti
Visita a friborgo

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
In ritardo di 8 ore.

Programma di massima per la prossima giornata di lavoro
-Documentare il tut

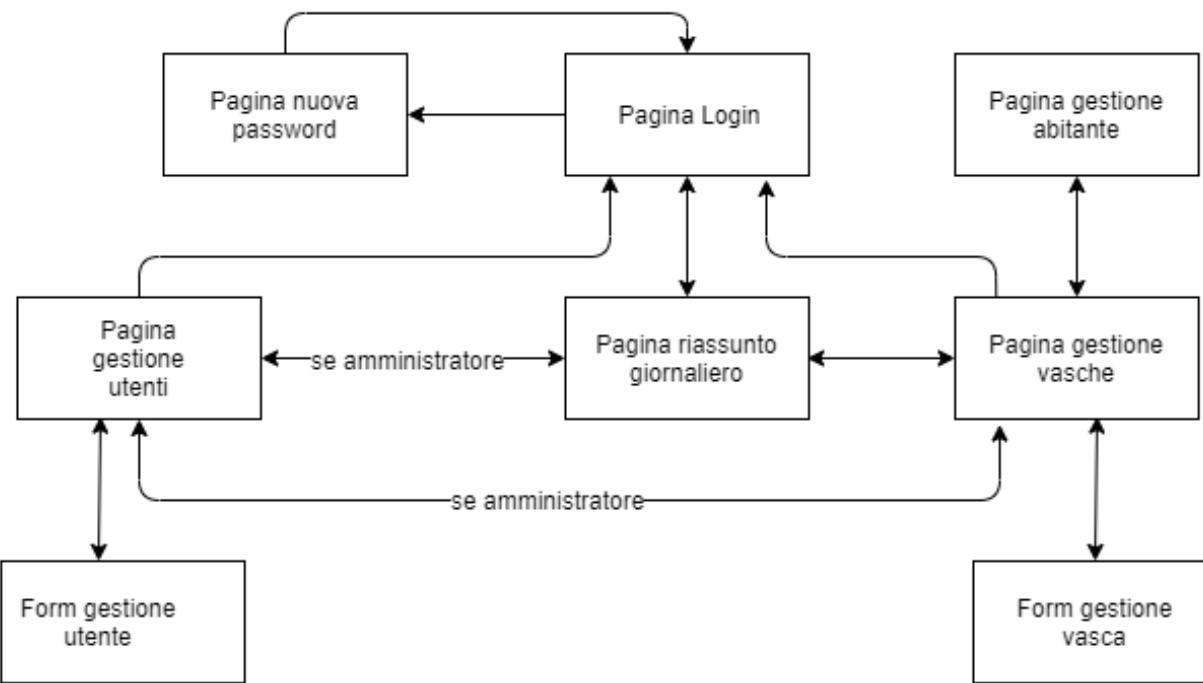
# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	06.12.2019

## Lavori svolti

Oggi ho documentato, quindi ho riguardato tutto quello che ho già fatto, ho chiarito alcuni dubbi con il docente Valsangiacomo per quanto riguarda ciò che devo e non devo modificare di quello che ho già fatto poi ho fatto il nuovo schema del database aggiornato, ho utilizzato un nuovo tool online che si chiama dbdiagram.io che permette di fare i diagrammi del database più chiari e con i tipi di dati visibili. Poi ho incominciato a documentare il frontend, nel farlo mi sono accorto che il campo data che rappresenta l'ultima volta che è stato effettuato il cambio d'acqua non mi stampa più la data correttamente ma me ne sono accorto troppo tardi e non sono riuscito a sistemarlo.

Ho fatto il design dell'architettura aggiornato, non ho tolto l'altro ma come per il database nel capitolo dell'implementazione mostro i cambiamenti rispetto la progettazione.



Nuovo diagramma del database.

vasca		abitante		utente	
nome	varchar(45)	specie	varchar(45)	email	varchar(255)
magnesio	int(4)	genere	varchar(15)	nome	varchar(45)
calcio	int(3)	nome_vasca	varchar(45)	cognome	varchar(45)
kh	int(2)	tipo	varchar(10)	tipo	varchar(45)
ultimo_cambio_acqua	date	numero	int(4)	numeroTelefonico	varchar(20)
litri	int(7)			cambioPassword	tinyint(1)
				password	varchar(255)

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

#### Programma di massima per la prossima giornata di lavoro

-Documentare il tutto

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	10.12.2019

## Lavori svolti

Oggi ho risolto il problema della data semplicemente perché nel data base avevo messo int al posto di date si vede che l'ultima volta toccando ho l'ho modificato per sbaglio.

Poi ho definito delle costanti per tutte le variabili che ri ripetevano nel file di config, in questo modo se voglio cambiare il valore è più facile e più pulito. Ho sistemato il codice la dove potevo, ottimizzato e commentato e poi ho continuato a fare la documnetazione. Ho parlato con il docente Valsangiaomo e il mio formatore 2 minuti poi ho fatto alcune domande al mio formatore sulla doc.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

## Programma di massima per la prossima giornata di lavoro

-Documentare il tutto

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	12.12.2019

## Lavori svolti

Oggi ho documentato l'implementazione, non ancora finito ma quasi, nel farlo ho sistemato anche un po' il codice. Poi sono ritornato sul capitolo 3.4 dove dovrei fare il design procedurale del sito web e li mi sono bloccato perché non so bene cosa mettere e cosa fare, per il momento l'ho saltato lo farò per ultimo.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

## Programma di massima per la prossima giornata di lavoro

-Documentare il tutto

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	13.12.2019

## Lavori svolti

Oggi ho continuato a documentare l'implementazione e ho quasi finito, alla fine ho deciso di separare la parte del backend in model e controller dove descrivo i metodi principali. All'inizio pensavo di documentare i controller e i model assieme ma usciva troppo confuso quindi ho separato. Comunque documento tutto tranne i metodi index e i costruttori e gli attributi.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

## Programma di massima per la prossima giornata di lavoro

-Documentare il tutto

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	17.12.2019

## Lavori svolti

Oggi ho fatto l'autovalutazione, ho completato la documentazione dell'implementazione e i test case.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

## Programma di massima per la prossima giornata di lavoro

-Documentare il tutto

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	19.12.2019

## Lavori svolti

Oggi ho finito la documentazione ho riguardato un po' tutto, codice e documentazione.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In ritardo di 8 ore.

## Programma di massima per la prossima giornata di lavoro

- Stampare tutto
- Guardare ultime cose

# Diario di lavoro

Luogo	Arti Mestieri Trevano
Data	19.12.2019

## Lavori svolti

Consegna progetto.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

-

## Programma di massima per la prossima giornata di lavoro

-

```
DROP EVENT IF EXISTS updateValueTank;

DELIMITER $$$
CREATE EVENT updateValueTank ON SCHEDULE EVERY 7 DAY DO BEGIN
    DECLARE nomeVasca varchar(50);
    DECLARE calcioVasca INT;
    DECLARE magnesioVasca INT;
    DECLARE khVasca INT;

    DECLARE cur CURSOR FOR SELECT nome FROM vasca;

    OPEN cur;
    read_loop: LOOP
        FETCH NEXT FROM cur INTO nomeVasca;
        select magnesio into magnesioVasca from vasca where nome=nomeVasca;
        select calcio into calcioVasca from vasca where nome=nomeVasca;
        select kh into khVasca from vasca where nome=nomeVasca;
        UPDATE vasca SET kh=(khVasca-1),calcio=(calcioVasca-
25),magnesio=(magnesioVasca-100) WHERE (nome=nomeVasca);

    END LOOP;
    CLOSE cur;
END $$$
DELIMITER ;
```

Mattia Ruberto I4AA Progetto Gestione Acquari Marini Database

```
drop database GestioneAcquariMariniProva;
create database GestioneAcquariMariniProva;
use GestioneAcquariMariniProva;

create table utente(
    email varchar(255) primary key not null,
    nome varchar(45),
    cognome varchar(45),
    tipo varchar(45),
    numeroTelefonico varchar(20),
    cambioPassword tinyint(1),
    password varchar(255) not null
);

create table vasca (
    nome varchar(45) primary key not null,
    calcio INT(11),
    magnesio INT(11),
    kh INT(11),
    `ultimo cambio acqua` DATE,
    Litraggio INT(11)
);

CREATE TABLE abitante (
    specie VARCHAR(45),
    genere VARCHAR(15),
    tipo VARCHAR(45),
    nome_vasca VARCHAR(45),
    numero INT(11),
    PRIMARY KEY (specie,genere),
    FOREIGN KEY (nome_vasca) REFERENCES vasca(nome)
);
INSERT INTO utente (email, nome, cognome, tipo, numeroTelefonico, cambioPassword, password) VALUES ('gestioneacquarimarin@gmail.com', 'Administrator', 'Administrator', 'Admin', '+41 79 232 32 33', '1', '$2y$10$V/7rIxZQPzNEnN34KM5u7eIr/e9xAiMuzSw.eMUODEY8cDsxevYYy');
```