

# Diario di lavoro

Luogo	Canobbio
Data	03.09.2019

## Lavori svolti

Durante questa giornata abbiamo ricevuto la lista dei progetti di semestre disponibili e dopo averli letti tutti abbiamo dovuto scegliere quello da fare. Io ho scelto il progetto del docente Guido Montalbetti riguardante lo sviluppo di un applicativo per la gestione dei parcheggi.

In seguito ho stilato una lista di domande da fare al docente riguardanti dei punti a me poco chiari nella spiegazione del progetto nel QdC.

Ho creato il repository su GitHub perchè lo userò come cloud assieme a OneDrive.

Dopo l'orario scolastico sono andato dal docente per sottoporgli le domande precedentemente stilate.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

-

## Programma di massima per la prossima giornata di lavoro

Fare il Gantt preventivo.

# Diario di lavoro

Luogo	Canobbio
Data	05.09.2019

## Lavori svolti

Durante questa giornata mi sono occupato della stesura del Gantt preventivo.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Preparare l'ambiente di lavoro e iniziare la stesura dei requisiti.

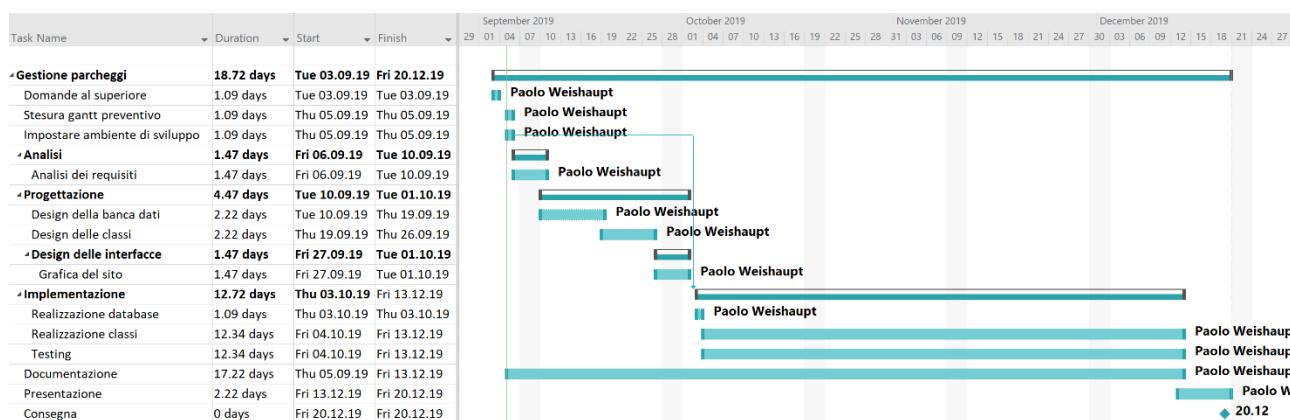


Figura 1 Gantt Preventivo

# Diario di lavoro

Luogo	Canobbio
Data	06.09.2019

## Lavori svolti

Durante questa giornata ho dapprima modificato il Gantt preventivo che avevo stilato durante la giornata di ieri perché avevo dei problemi con lo scheduling, in seguito mi sono rimesso in pari con la programmazione preparando l'ambiente di sviluppo e iniziando la stesura dai requisiti. Inoltre, ho fatto il capitolo 1.1 della documentazione.

Come ambiente di sviluppo ho scelto Windows 10 con il software XAMPP in versione 7.3.0-0.

## Problemi riscontrati e soluzioni adottate

Dopo che il docente Valsangiacomo ci ha spiegato come impostare gli orari, mi sono accorto che lo avevo fatto male quindi ho creato un nuovo foglio di lavoro con gli orari aggiornati.

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Continuare con la stesura dei requisiti.

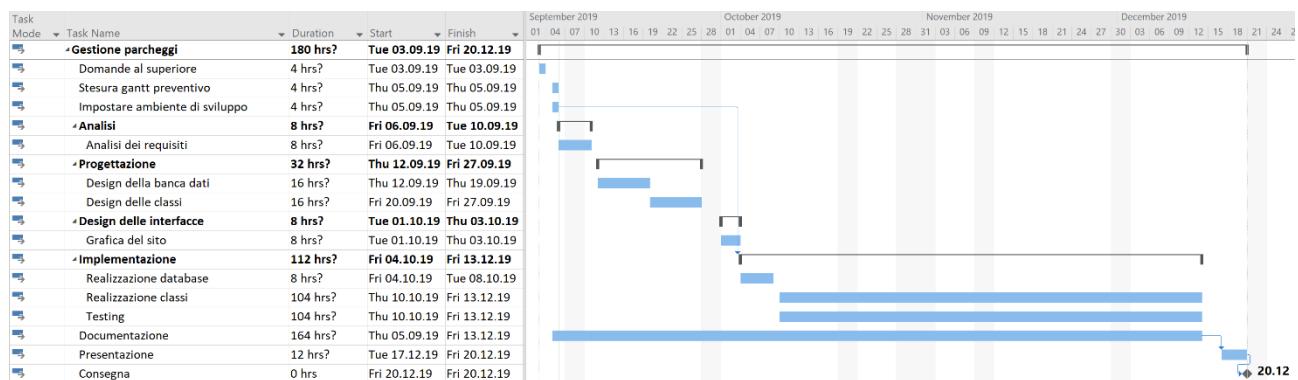


Figura 1 Nuovo Gantt preventivo

# Diario di lavoro

Luogo	Canobbio
Data	10.09.2019

## Lavori svolti

Durante questa giornata di lavoro ho finito con la stesura dei requisiti. Nelle seconde ore con il docente Raimondi abbiamo risolto il problema del proxy che non faceva pushare su GitHub e ho iniziato a progettare la banca dati.

Per la documentazione ho finito i capitoli 2.2, 2.4 e 2.5.2.

## Problemi riscontrati e soluzioni adottate

Il proxy non ci permetteva di pushare su GitHub quindi il docente Raimondi ha preso gli indirizzi IP di tutti e li ha messi in una whitelist per quello che ho capito. La prossima lezione invece prenderà gli indirizzi MAC di tutti e li sostituirà agli IP

## Punto della situazione rispetto alla pianificazione

In anticipo.

## Programma di massima per la prossima giornata di lavoro

Continuare con la progettazione della banca dati.

ID: REQ-001	
Nome	Creazione applicativo web
Priorità	1
Versione	1.0
Note	Il progetto vuole creare un sito Web dove poter gestire i parcheggi disponibili della scuola
Sotto requisiti	
001	Le persone potranno ricercare i posteggi disponibili
002	Le persone potranno riservare i posteggi
003	Le persone potranno stampare la loro riservazione

ID: REQ-002	
<b>Nome</b>	Creazione maschera admin
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Il sito dovrà avere una parte del menù riservata agli amministratori
<b>Sotto requisiti</b>	
<b>001</b>	Gli amministratori potranno inserire i parcheggi disponibili
<b>002</b>	Gli amministratori potranno gestire le persone registrate al sito

ID: REQ-003	
<b>Nome</b>	Funzionalità principali per gli utenti
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Il sito dovrà avere due sezioni principali: Offerta e Ricerca
<b>Sotto requisiti</b>	
<b>001</b>	Nella sezione Offerta l'utente dovrà essere in grado di mettere a disposizione il suo parcheggio.
<b>002</b>	Nella sezione Offerta le persone potranno inserire le date della disponibilità del loro posto auto (mattina, pomeriggio o tutto il giorno) e in più indicare il loro numero di targa oltre ai loro dati personali di base
<b>003</b>	Nella sezione "Ricerca" si potranno ricercare i parcheggi a disposizione inserendo come parametro il giorno preciso o anche un periodo, p.es. dal 2.12.2019 al 5.12.2019
<b>004</b>	Sulla lista dei risultati, fornita dalla ricerca, si potranno ordinare e filtrare i dati

ID: REQ-004	
<b>Nome</b>	Creazione maschera di login
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Le persone dovranno registrarsi al sito ed inserire i propri dati principali per poter riservare il posto auto
<b>Sotto requisiti</b>	
001	I campi minimi da compilare saranno: nome, cognome, email e un recapito telefonico
002	L'applicativo dovrà prevedere un meccanismo di conferma della registrazione per l'identificazione della persona, p.es. l'invio di un link all'indirizzo di posta elettronica della persona, impostato in fase di registrazione

ID: REQ-005	
<b>Nome</b>	Specifiche per la riservazione
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Di seguito le specifiche da seguire per la riservazione di un posteggio
<b>Sotto requisiti</b>	
001	Il costo del posto auto è di 10 CHF al giorno da conteggiare e fatturare mensilmente (mezza giornata sono 5 CHF)
002	Al fronte di una riservazione, le persone riceveranno una conferma tramite email
003	Prevedere un meccanismo di concorrenzialità per evitare che una persona in procinto di riservare il parcheggio, si veda portar la sua riservazione da un'altra persona

ID: REQ-006	
<b>Nome</b>	Gestione stampe
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Di seguito le specifiche da seguire per la gestione delle stampe
<b>Sotto requisiti</b>	
<b>001</b>	Gli amministratori potranno stampare la situazione dei parcheggi per un determinato periodo (a disposizione, riservati)
<b>002</b>	Gli amministratori potranno stampare la fattura per una persona singola o per tutte le persone, scegliendo un periodo da loro impostato, generalmente una stampa mensile. Il saldo della fattura è di 30 giorni netto
<b>003</b>	Le persone che avranno riservato il parcheggio potranno stampare la loro riservazione
<b>004</b>	Gli amministratori potranno stampare i richiami per le persone che non hanno ancora saldato la loro riservazione. A metà di ogni mese, giorno feriale, verranno stampati i richiami delle fatture. Prevedere un meccanismo automatico con notifica all'utente prima della stampa

ID: REQ-007	
<b>Nome</b>	Gestione fatturazione
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Di seguito le specifiche da seguire per la gestione delle fatturazioni
<b>Sotto requisiti</b>	
<b>001</b>	Se dopo il primo e unico richiamo la persona non avesse ancora saldato la fattura, la persona sarà "bloccata" e non avrà più la possibilità di riservare i parcheggi sino al saldo della fattura. In questo caso la persona interessata sarà avvisata via email del suo blocco e del richiamo non ancora saldato

ID: REQ-008	
Nome	Convalida dati
Priorità	1
Versione	1.0
Note	L'applicativo avrà una convalida dei dati
<b>Sotto requisiti</b>	
001	La convalida dei dati immessi è una funzionalità dell'applicativo, p.es. l'email e il numero di telefono devono avere un formato corretto

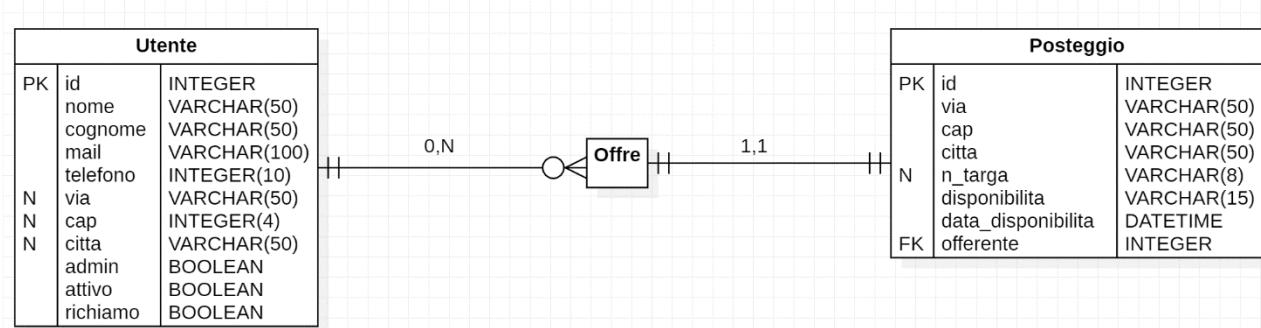


Figura 1 Design database

# Diario di lavoro

Luogo	Canobbio
Data	12.09.2019

## Lavori svolti

Durante questa giornata di lavoro ho continuato con la stesura del design della banca dati. Matteo Forni mi ha aiutato nella stesura di alcune parti del diagramma.

Ho iniziato la stesura del Gantt consuntivo.

Domanda al supervisore: Una persona può avere più di posteggi da offrire?

Risposta del supervisore: No.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Continuare con la progettazione della banca dati e discuterne con il supervisore.

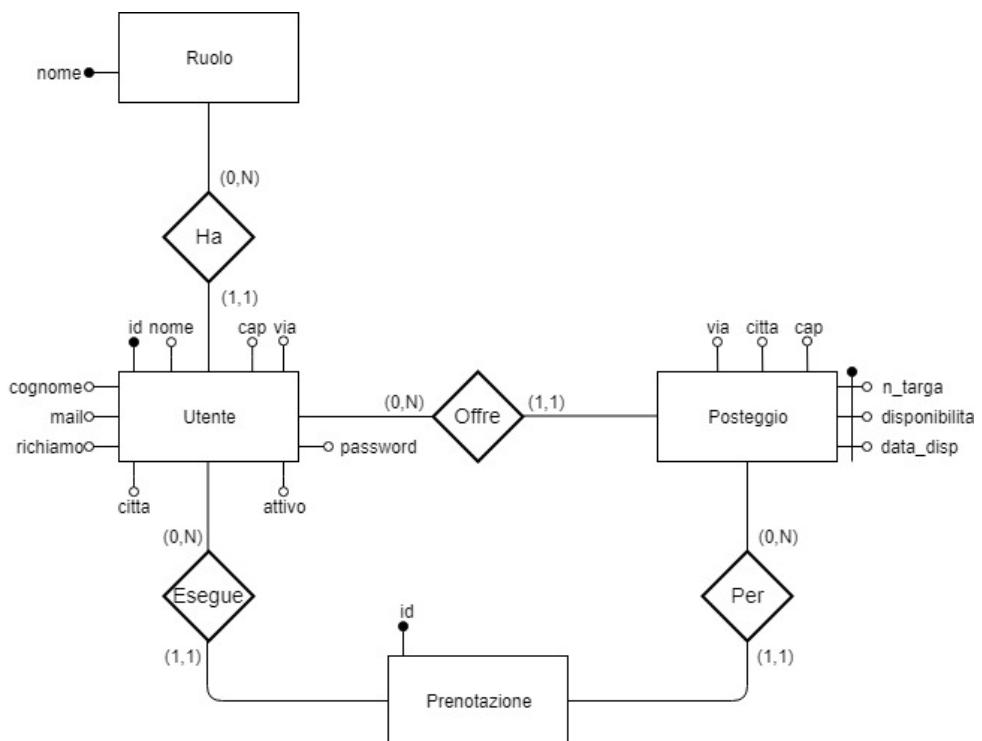


Figura 1 Design del database

# Diario di lavoro

Luogo	Canobbio
Data	13.09.2019

## Lavori svolti

Durante questa giornata di lavoro ho finito la stesura del design del database in base alla discussione avuta con il committente e ho iniziato con la sua implementazione.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In anticipo.

## Programma di massima per la prossima giornata di lavoro

La prossima giornata lavorativa sarò assente.

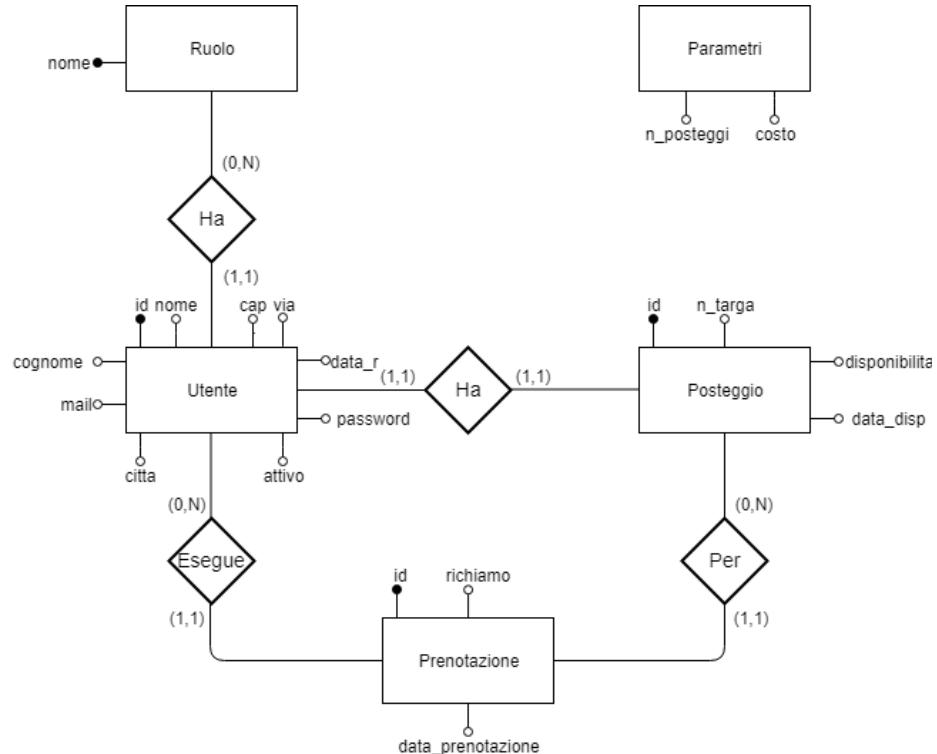
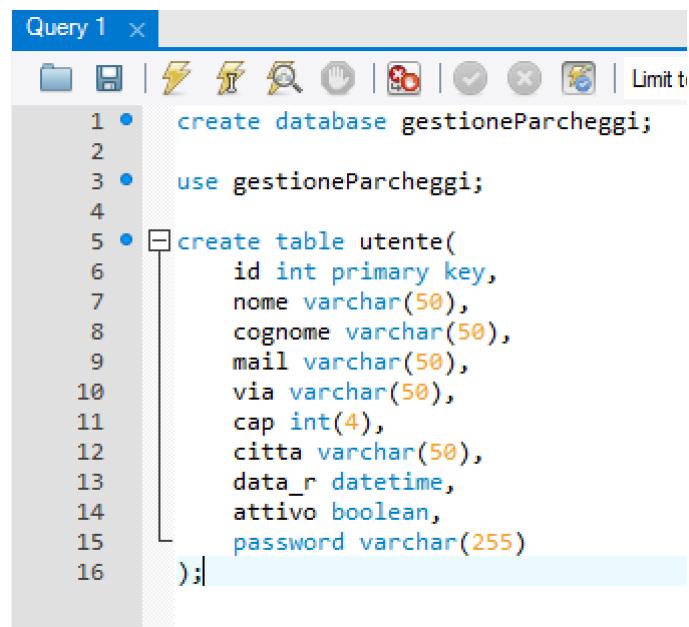


Figura 1 Design del database



The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The code in the editor is as follows:

```
1 •  create database gestioneParcheggi;
2
3 •  use gestioneParcheggi;
4
5 •  create table utente(
6      id int primary key,
7      nome varchar(50),
8      cognome varchar(50),
9      mail varchar(50),
10     via varchar(50),
11     cap int(4),
12     citta varchar(50),
13     data_r datetime,
14     attivo boolean,
15     password varchar(255)
16 );
```

The code creates a database named "gestioneParcheggi" and selects it. It then creates a table "utente" with various fields including primary key "id", string fields for name, surname, email, address, and city, an integer field for zip code, a date/time field for registration date, a boolean field for active status, and a password field.

# Diario di lavoro

Luogo	Canobbio
Data	19.09.2019

## Lavori svolti

Durante questa giornata di lavoro ho migliorato il design del database finale e ho creato il database.

## Problemi riscontrati e soluzioni adottate

La vm che uso per il progetto ha smesso di funzionare e dopo vari tentativi di ripristino ho dovuto ricrearne una da zero.

## Punto della situazione rispetto alla pianificazione

In anticipo.

## Programma di massima per la prossima giornata di lavoro

Continuerò la documentazione.

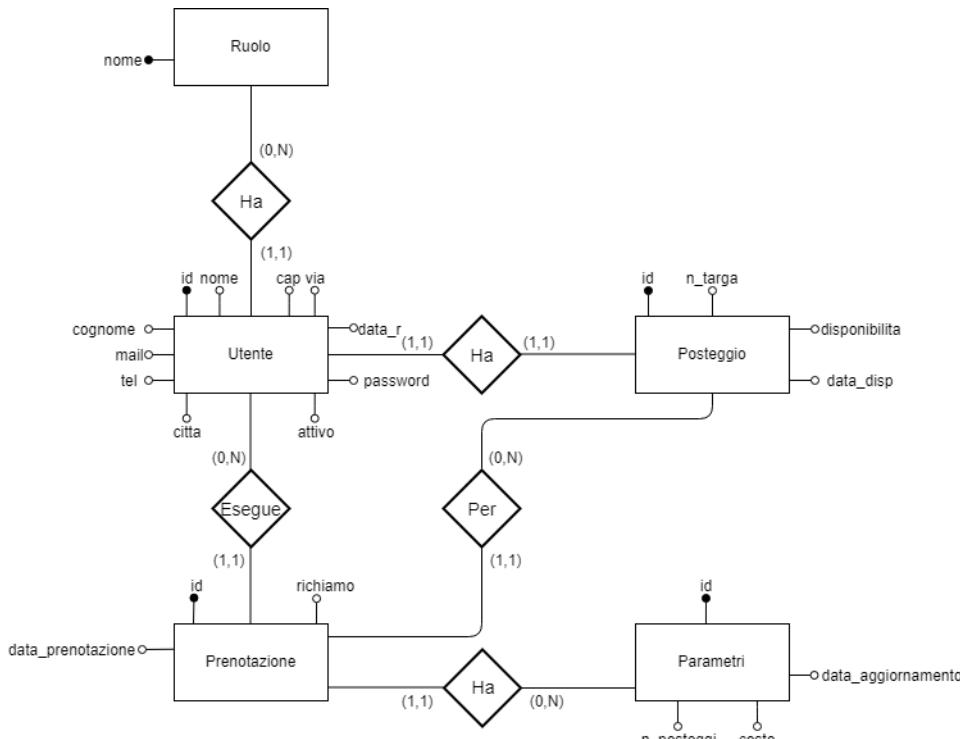


Figura 1 Design del database

```
create database gestione_parcheggi;

use gestione_parcheggi;

create table parametri(
    id int primary key,
    n_posteggi int,
    costo int,
    data_aggiornamento datetime
);
create table ruolo(
    nome varchar(50) primary key,
);
create table utente(
    id int primary key,
    ruolo varchar(50),
    nome varchar(50) not null,
    cognome varchar(50)not null,
    mail varchar(50) not null,
    via varchar(50),
    cap int(4),
    citta varchar(50),
    tel int(10) not null,
    data_r datetime,
    attivo boolean,
    password varchar(255),
    foreign key(ruolo) references ruolo(nome)
);
create table posteggio(
    id int primary key,
    disponibilita varchar(50),
    data_disp datetime,
    n_targa varchar(8),
    id_utente int,
    foreign key(id_utente) references utente(id)
);
create table prenotazione(
    id int primary key,
    richiamo boolean,
    data_prenotazione datetime,
    id_utente int,
    id_posteggio int,
    id_parametri int,
    foreign key(id_utente) references utente(id),
    foreign key(id_posteggio) references posteggio(id),
    foreign key(id_parametri) references parametri(id),
);
```

# Diario di lavoro

Luogo	Canobbio
Data	20.09.2019

## Lavori svolti

Nello scorso diario mi sono accorto di aver fatto un errore. Per “ho creato il database” intendeva dire che avevo finito di scrivere il codice.

Oggi ho infatti fatto partire il codice e il database è stato creato senza problemi.

Mi sono occupato dei capitoli della documentazione 2.5.1, 3.2.1, 3.2.2 e 3.2.3.

## Problemi riscontrati e soluzioni adottate

Ho avuto nuovamente problemi con la vm del progetto su Windows. Ho risolto provando a farla partire da linux (distro Deepin 15.11) che ho in dual boot sul mio computer.

## Punto della situazione rispetto alla pianificazione

In anticipo.

## Programma di massima per la prossima giornata di lavoro

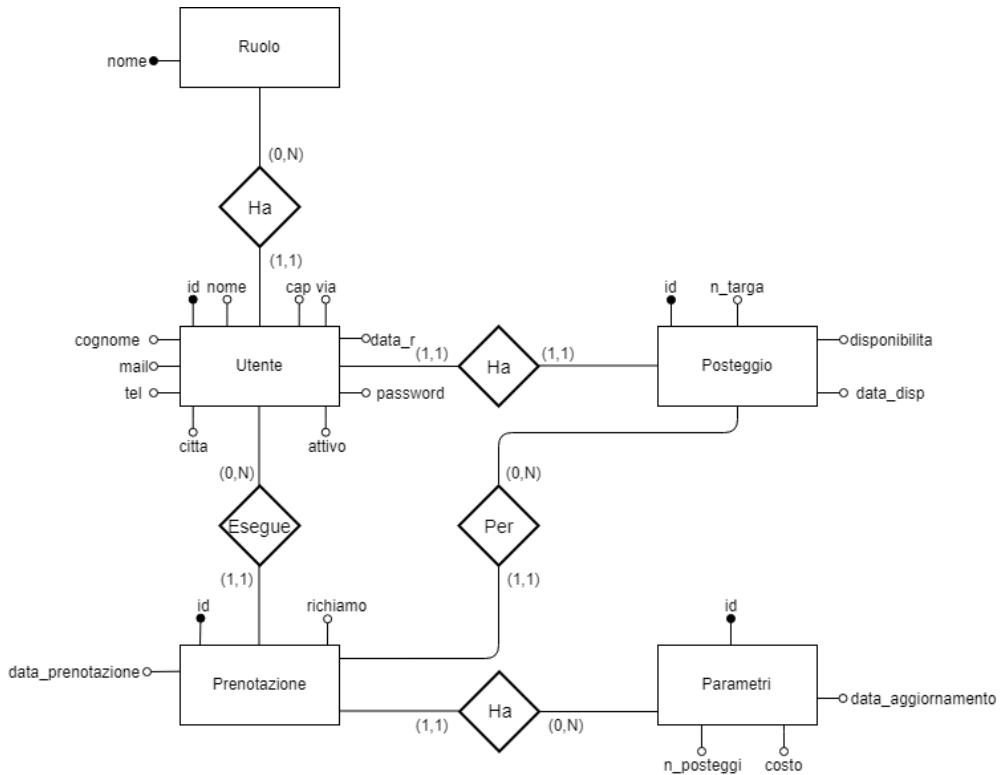
Continuerò la documentazione e inizierò con il design delle interfacce.

### 2.5.1 Software

I software utilizzati per la realizzazione di questo progetto sono:

- Microsoft Word 2016
- Visual Studio Code
- Project
- Google Chrome
- VMWare Workstation
- XAMPP 7.3.0-0
- MySQL Workbench 8.0 CE
- draw.io online
- SourceTree

### 3.2.1 Diagramma E-R



### 3.2.2 Schema procedurale

Parametri(id, n\_posteggi, costo, data\_aggiornamento)

Ruolo(nome)

Utente(id, ruolo(FK), nome, cognome, mail, via, cap, citta, tel, data\_r, attivo, password)

Posteggio(id, id\_utente(FK), disponibilita, data\_disp, n\_targa)

Prenotazione(id, id\_utente(FK), id\_posteggio(FK), id\_parametri(FK), richiamo, data\_prenotazione)

#### 1.1.1 Descrizione database

Il database è composto da 5 entità.

L'entità PARAMETRI serve per contenere il numero di parcheggi disponibili, il costo di affitto di un parcheggio e la data dell'ultimo aggiornamento del costo.

L'entità RUOLO contiene solo un campo ed è il nome del ruolo. Questo perché se in futuro si dovesse avere il bisogno di cambiare i nomi dei ruoli non si dovrà cambiare manualmente per tutti gli utenti.

L'entità UTENTE rappresenta gli utenti del db e contiene le loro informazioni. L'attributo attivo serve per la verifica dell'e-mail perché finché non viene verificata il suo valore sarà su false e l'account sarà disabilitato. L'attributo potrà inoltre essere settato a false se l'utente dovesse avere dei richiami su una fattura.

L'entità POSTEGGIO rappresenta un posteggio e contiene i dati necessari al suo riconoscimento. Gli attributi disponibilita, data\_disp e n\_targa verranno utilizzati se il parcheggio dovrà essere messo in offerta.

L'entità PRENOTAZIONE serve per gestire le prenotazioni degli utenti di un parcheggio. L'attributo richiamo ha di default il valore false. Se la fattura riguardante quella prenotazione non dovesse essere saldata verrà settato a true e al prossimo richiamo l'utente verrà disabilitato.

# Diario di lavoro

Luogo	Canobbio
Data	24.09.2019

## Lavori svolti

Oggi per la maggior parte del tempo mi sono occupato della stesura del design delle interfacce. Verso fine lezione ho iniziato a cercare un framework per creare il sito e ho trovato MDBBootstrap.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In anticipo.

## Programma di massima per la prossima giornata di lavoro

Aggiungerò alla documentazione il capitolo sul design delle interfacce e inizierò a implementare il sito.

# Diario di lavoro

Luogo	Canobbio
Data	26.09.2019

## Lavori svolti

Oggi ho iniziato con l'implementazione delle interfacce riuscendo a finire la pagina di login e la homepage del sito.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In anticipo.

## Programma di massima per la prossima giornata di lavoro

Aggiungerò alla documentazione il capitolo sul design delle interfacce e continuare l'implementazione delle interfacce.

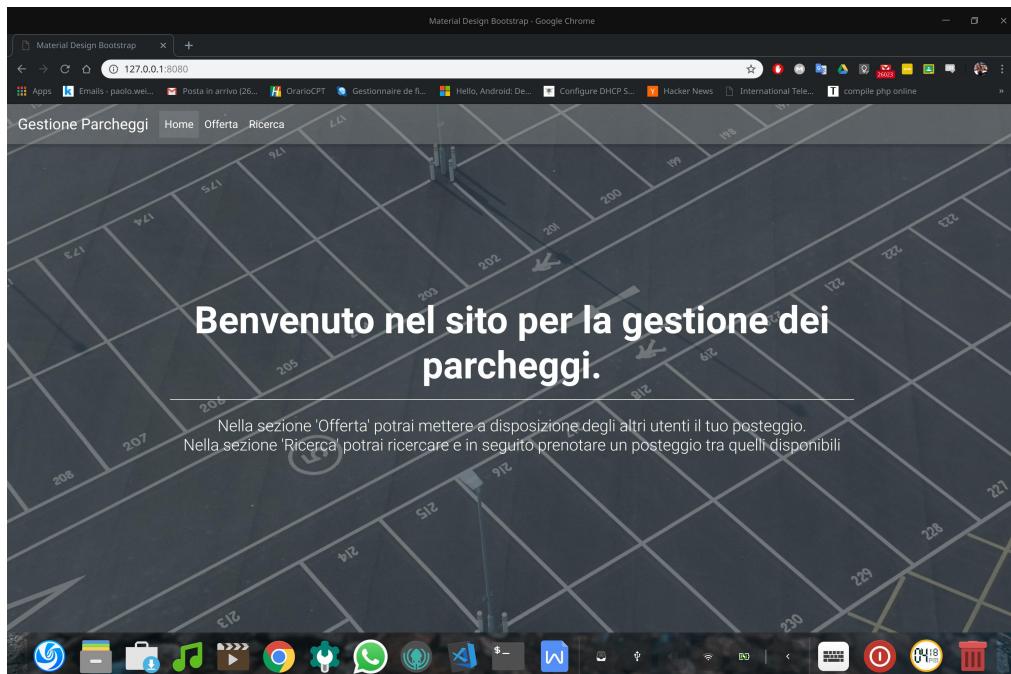


Image 1 Homepage del sito

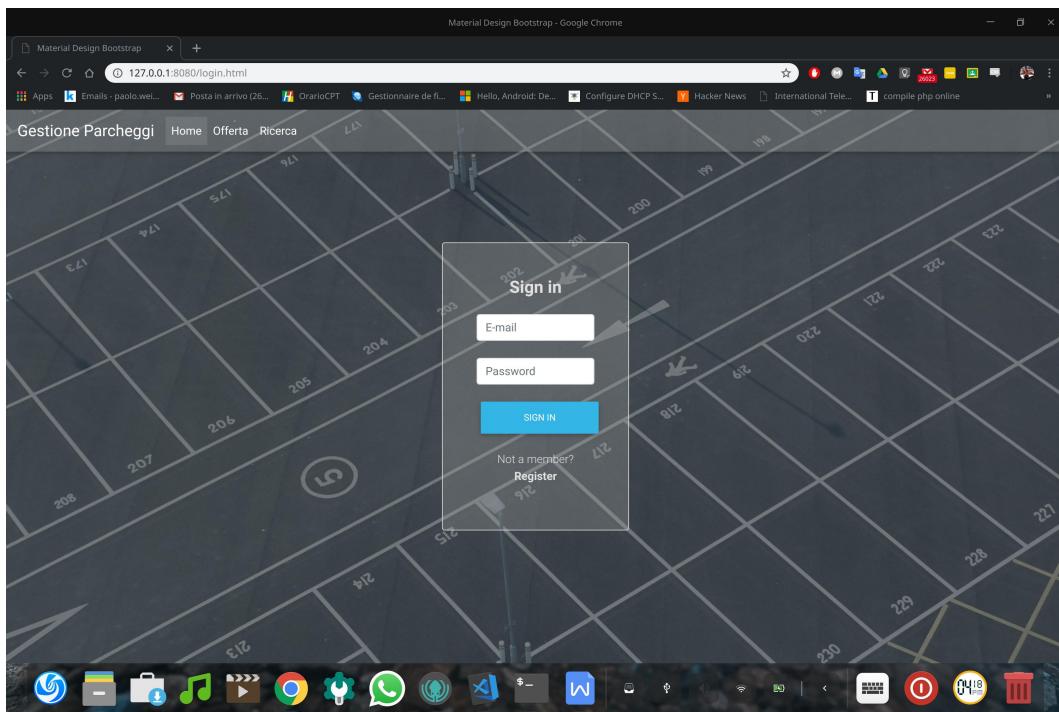


Image 2 Login

# Diario di lavoro

---

Luogo	Canobbio
Data	27.09.2019

**Lavori svolti**

Oggi ho messo il lavoro che ho fatto ieri in un template MVC.  
Dalle 15:00 alle 15:45 circa ho avuto un colloquio con il Sig. Togni dell'Ufficio Lingue e Stage all'Estero.

**Problemi riscontrati e soluzioni adottate****Punto della situazione rispetto alla pianificazione**

In anticipo.

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

Continuerò con lo sviluppo del sito.

# Diario di lavoro

---

Luogo	Canobbio
Data	01.10.2019

**Lavori svolti**

Ho continuato a implementare il codice nel template MVC e ho aggiunto il collegamento dalla homepage alla pagina di login.

**Problemi riscontrati e soluzioni adottate****Punto della situazione rispetto alla pianificazione**

In anticipo.

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

Continuerò con lo sviluppo del sito.

# Diario di lavoro

---

Luogo	Canobbio
Data	03.10.2019

## Lavori svolti

In questa giornata di lavoro ho trasferito il sito su un altro template MVC. Il nuovo template è stato sviluppato da Filippo Finke e ho deciso di usarlo perchè era più intuitivo di quello vecchio.

Inoltre ho implementato le funzionalità di login e logout.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In anticipo.

## Programma di massima per la prossima giornata di lavoro

Continuerò con lo sviluppo del sito.

# Diario di lavoro

---

Luogo	Canobbio
Data	04.10.2019

## Lavori svolti

Durante la giornata di oggi ho migliorato il sistema di login mostrando in alto a destra il nome dell'utente loggato. Ho implementato la funzione che mostra il riquadro per la dashboard admin se l'utente loggato è un amministratore.

Ho modificato la pagina di errore 404 in modo da applicarle lo stesso tema del sito.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Continuerò con lo sviluppo del sito.

# Diario di lavoro

Luogo	Canobbio
Data	08.10.2019

## Lavori svolti

Oggi ho fatto in modo che se un utente è loggato non gli venga mostrato il pulsante di login, ma solo quello di logout. Viceversa, ho fatto in modo che un utente non loggato veda solo il pulsante di login. Per fare ciò mi è ritornata molto utile la seguente risposta su GitHub:

<https://stackoverflow.com/questions/32398637/running-php-inside-of-php-echo#>

Ho caricato il database sul server.

Ho finito il layout della pagina di offerta del posteggio.

## Problemi riscontrati e soluzioni adottate

Ho avuto un problema con il PHP che eseguivo dentro a un echo. Nella barra degli indirizzi stampava il seguente indirizzo:

<http://127.0.0.1:8080/home/%3C?php%20echo%20URL.home/index;%20?%3E>

Ho risolto grazie alla risposta trovata su GitHub che ho citato sopra.

Codice precedente:

```
'<a class="dropdown-item" href="'.URL.'login/index'; ?>">Login</a>'
```

Codice fixato:

```
'<a class="dropdown-item" href="'.URL.'login/logout'.'">Logout</a>'
```

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Continuerò con lo sviluppo del sito.

# Diario di lavoro

---

Luogo	Canobbio
Data	10.10.2019

**Lavori svolti**

Oggi ho aggiunto la validazione dei dati inseriti nella pagina di login e in quella dell'offerta. In quella dell'offerta inoltre ho aggiunto la funzionalità che reimmette i dati precedentemente inseriti negli appositi input.

**Problemi riscontrati e soluzioni adottate****Punto della situazione rispetto alla pianificazione**

In linea.

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

Continuerò con lo sviluppo del sito.

# Diario di lavoro

Luogo	Canobbio
Data	11.10.2019

## Lavori svolti

Durante le prime due ore di questa giornata il docente Valsangiacomo ci ha spiegato come si svolgeranno gli LPI.

Dopo la pausa i docenti Montalbetti e Valsangiacomo hanno caricato su Moodle una guida per collegarsi al server FTP sul quale devo caricare il progetto. Il client utilizzato è FileZilla in versione 3.24.0. Ho caricato il sito sul server e ho aggiornato all'ultima versione il codice su GitHub.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Continuerò con lo sviluppo del sito.

# Diario di lavoro

Luogo	Canobbio
Data	15.10.2019

## Lavori svolti

Ho modificato la struttura delle tabelle utente e posteggio. Nella tabella utente ho creato il campo per salvare l'id del posteggio di un utente, mentre ho tolto il riferimento all'utente nel posteggio. Questo perchè non avevo pensato a come accedere al posteggio di un utente senza una prenotazione prenotazione. Ho aggiornato in seguito il capitolo della documentazione 3.2.2.

Da circa un mese ho cambiato il sistema operativo che uso giornalmente. Sono passato da Windows 10 a Deepin 15.11 basato su Debian 9 Stretch. Ho quindi aggiornato i capitoli 2.5.1 e 2.5.2. inoltre non userò più la virtual machine con Windows 10, ma testerò il tutto sulla mia macchina.

Ho iniziato lo sviluppo della pagina per la registrazione.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Continuerò con lo sviluppo del sito.

# Diario di lavoro

---

Luogo	Canobbio
Data	17.10.2019

## Lavori svolti

Oggi ho continuato con lo sviluppo della pagina di registrazione. La grafica è stata completata. Ho aggiunto i vari validator all'apposita classe.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Sviluppare la logica della registrazione e l'inserimento dei dati nel db.

# Diario di lavoro

Luogo	Canobbio
Data	18.10.2019

## Lavori svolti

Ho finito la logica della registrazione e l'inserimento dei dati nel db.

Ho avuto un colloquio con il docente Montalbetti durante il quale abbiamo discusso di come sta procedendo il progetto. Mi ha fatto notare delle mie sviste che si potranno leggere nel file apposito nella cartella Colloqui/18/10/2019.

Ho modificato il modo in cui notifico l'utente se ci sono degli errori nei form di login, offerta e registrazione. Ora uso la libraria Notify.js.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Risolvere i punti discussi durante il colloquio e salvare i dati dell'offerta sul database.

# Diario di lavoro

---

Luogo	Canobbio
Data	22.10.2019

## Lavori svolti

Nella giornata di oggi ho aggiunto dei dati in più al database. Le tabelle *parametri* e *ruolo* sono rimaste invariate mentre alle tabelle *posteggio* e *utente* ho aggiunto 1000 insert cadauna.

Ho trovato una libreria che mi permetteva di modificare manualmente il formato del picker per la data e l'ho implementata.

La libreria è la seguente:

<https://bootstrap-datepicker.readthedocs.io/en/latest/index.html>

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Modificare il gantt e iniziare il salvataggio dell'offerta.

# Diario di lavoro

Luogo	Canobbio
Data	24.10.2019

## Lavori svolti

Nella giornata di oggi ho finito lo sviluppo del salvataggio dell'offerta.

Ho chiesto tramite mail al docente Montalbetti se devo modificare il Gantt preventivo o consuntivo.

Ho iniziato lo sviluppo della pagina di ricerca dei posteggi. Per ora ho fatto solo il design.

## Problemi riscontrati e soluzioni adottate

La pagina non è scrollabile. Devo ancora trovare una soluzione

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Finire lo styling della tabella di ricerca.

# Diario di lavoro

Luogo	Canobbio
Data	25.10.2019

## Lavori svolti

Oggi ho risolto il problema della pagina di ricerca non scrollabile.

Ho avuto modo di discutere personalmente con il docente formatore. Per il Gantt devo copiare il preventivo e farne uno Intermedio.

Il sito sul server non è raggiungibile quindi il docente vedrà se riuscirà a farmi avere i log php per vedere le sue chiamate.

## Problemi riscontrati e soluzioni adottate

Ho aggirato il problema dello scroll spostando l'immagine di background e la maschera sul body e non sul div.

Non riesco a visualizzare il sito correttamente sul server.

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Aggiornare la versione del sito presente sull'hosting e continuare con lo sviluppo della pagina di ricerca.

# Diario di lavoro

Luogo	Canobbio
Data	05.11.2019

## Lavori svolti

Durante le vacanze ho provato a caricare il sito sull'hosting e da casa ha funzionato. Assieme a Filippo ho risolto il problema di visualizzazione del sito una volta caricato sull'hosting.

Oggi ho fatto il Gantt intermedio e ho risolto un problema di visualizzazione del datepicker.

## Problemi riscontrati e soluzioni adottate

La visualizzazione del sito sull'hosting non era corretta perchè nell'URL del file di config non usavo https, ma http.

I problemi di visualizzazione del datepicker erano dovuti al fatto che nel mio file di style personale avevo aggiunto uno style che rendeva il testo dei tr bianco non pensando che il datepicker fosse di base una tabella.

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Continuare lo sviluppo della pagina di ricerca.

# Diario di lavoro

Luogo	Canobbio
Data	07.11.2019

## Lavori svolti

Nella giornata di oggi mi sono occupato dello sviluppo della pagina di ricerca. Ho fatto in modo di ricevere i dati dal database e di mostrarli nella tabella.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

The screenshot shows a desktop application window titled "Gestione Parcheggi". The main view features a map of a parking area with various spots numbered (e.g., 205, 206, 207, 208, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228). Overlaid on the map is a table listing parking availability:

Disponibilità	Data disponibilità	Numero di targa	Prenota
Tutto il giorno	2019-11-20 00:00:00	GR-12345	BUTTON
Tutto il giorno	2019-11-23 00:00:00		BUTTON
Tutto il giorno	2019-11-07 00:00:00	TI-234234	BUTTON
Proprietario	Disponibilità	Data disponibilità	Prenota

The application interface includes a top navigation bar with "Gestione Parcheggi", "Home", "Offerta", and "Ricerca" buttons, and a user profile icon. The bottom taskbar displays various system icons.

Table 1 Pagina di ricerca

# Diario di lavoro

Luogo	Canobbio
Data	08.11.2019

## Lavori svolti

Durante la giornata di oggi ho avuto un colloquio con il docente formatore. I punti della discussione sono disponibili su GitHub nella cartella Colloqui.

In seguito, grazie all'aiuto di Filippo Finke, ho risolto un problema riguardante la pagina di ricerca. La data anche se formattata non veniva mostrata correttamente. Questo perché facevo le modifiche sull'oggetto row senza il riferimento alla variabile che passavo alla view. Ho risolto mettendo il carattere & davanti alla variabile row.

```
foreach(self::$parcheggi as &$row)
```

Ho risolto un problema che faceva apparire la notifica di errore agli utenti senza parcheggio ogni volta che entravano nella pagina di offerta. Ho fatto anche in modo che quando lo stesso utente senza parcheggio viene reindirizzato al form di offerta non gli vengano salvati i dati inseriti.

```
unset($_SESSION['selectedDate']);  
unset($_SESSION['carPlate']);  
ViewLoader::load( template: 'offerta/index');  
unset($_SESSION['noPark']);
```

Ho aggiunto i commenti a tutti i controllers.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Svolgere i lavori discussi durante il colloquio.

The screenshot shows a web browser window with a dark-themed interface. At the top, there are several tabs: 'GitKraken Position', 'Bootstrap 4 DataTables - example', 'Bootstrap table - styles & example', 'Mails - paolo.weishaupt@samtr...', and 'jquery - Change table header color'. The main content area has a header 'Gestione Parcheggi' with links for 'Home', 'Offerta', and 'Ricerca'. On the right, there is a user profile for 'Deena'. Below the header is a map of a parking lot with various numbered spots (e.g., 205, 206, 207, 208, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230). Overlaid on the map is a table with the following data:

Disponibilità	Data disponibilità	Numero di targa	Prenota
Tutto il giorno	2019-11-20 00:00:00	GR-12345	<button>BUTTON</button>
Tutto il giorno	2019-11-23 00:00:00		<button>BUTTON</button>
Tutto il giorno	2019-11-07 00:00:00	TI-234234	<button>BUTTON</button>
Proprietario	Disponibilità	Data disponibilità	Prenota

At the bottom of the screen, there is a dock with various application icons, including a terminal, file manager, browser, and system tray icons.

Table 1 Pagina di ricerca

# Diario di lavoro

Luogo	Canobbio
Data	12.11.2019

## Lavori svolti

Durante questa giornata lavorativa ho finito di commentare il codice finora prodotto.

```
1  <?php
2  /* Classe model per la gestione delle offerte. ...*/
3  namespace models;
4
5  use ...
6
7
8
9
10 class OffertaModel
11 {
12
13
14     /**
15      * @var Disponibilità. ...
16      */
17     private static $selectedVal;
18
19
20     /**
21      * @var Data della disponibilità. ...
22      */
23     private static $selectedDate;
24
25
26     /**
27      * @var Numero di targa. ...
28      */
29     private static $selectedCarPlate;
30
31
32     /**
33      * @var Query da eseguire. ...
34      */
35     private static $statement;
36
37
38     /**
39      * Funzione per l'aggiunta di un'offerta. ...
40      */
41     public static function addOfferta()
42     {
43         if(isset($_POST['offri'])){...}
44
45         if(isset($_SESSION['dateError']) || isset($_SESSION['carPlateError'])){...}
46
47         if(Users::hasParcheggio()){...}
48         else{...}
49     }
50
51 }
```

Table 1 Esempio di classe commentata.

Ho iniziato lo sviluppo della prenotazione di un parcheggio.

```
1 <?php
2 /**
3 * Classe model per la gestione delle prenotazioni.
4 */
5 namespace models;
6
7 use Libs\Database as Database;
8
9 class PrenotaModel
10 {
11     private $statement;
12
13     public function getParcheggioInfo($id_parcheggio)
14     {
15         $this->statement = Database::get()->prepare(
16             statement: "SELECT utente.nome, utente.cognome, utente.tel, parametri.costo
17             FROM utente, parametri, posteggio
18             WHERE utente.id_posteggio = posteggio.id;
19         ");
20
21         $this->statement->execute();
22     }
23 }
```

Table 2 Prima bozza della classe di prenotazione.

In questa prima bozza di codice ho definito la proprietà che userò per il prepared statement. Nella funzione seleziono in nome, il cognome e il numero di telefono dell'utente e la tariffa base per la prenotazione di un parcheggi dove l'id del parcheggio nella tabella di ricerca corrisponde a quello assegnato all'utente.

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In linea.

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

Continuare lo sviluppo del metodo di prenotazione.

# Diario di lavoro

Luogo	Canobbio
Data	14.11.2019

## Lavori svolti

Durante questa giornata lavorativa ho continuato lo sviluppo della prenotazione. Il risultato quasi finale è il seguente.

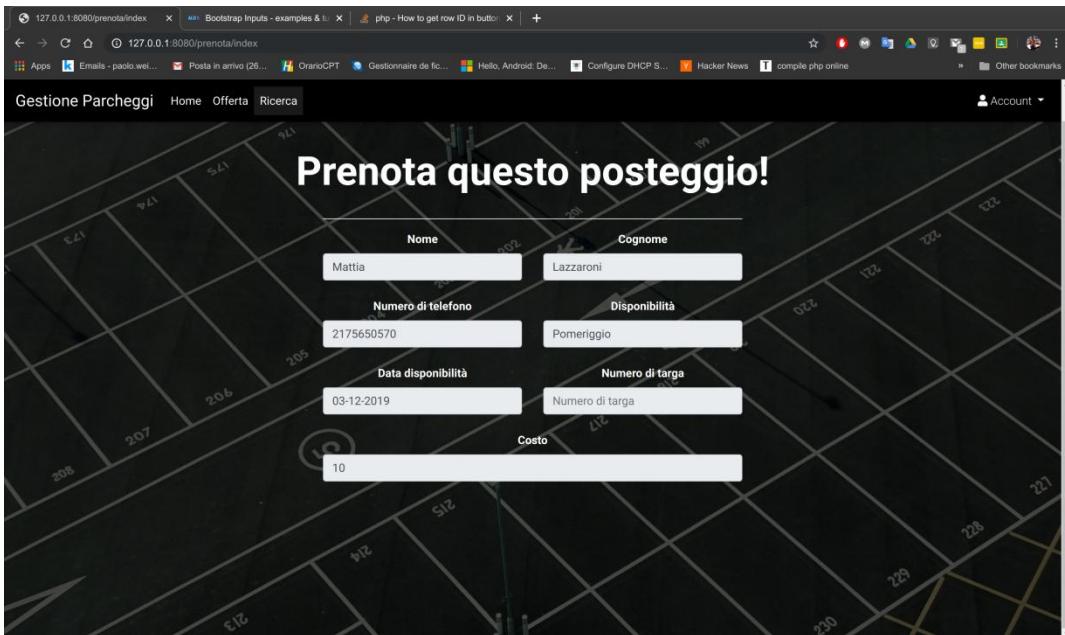


Table 1 Layout della pagina

Nella pagina di ricerca dei posteggi salvo l'id del posteggio in un input nascosto.

```
<?php foreach ($parcheggi as $row): ?>
<tr>
    <td><?php echo $row['disponibilita'] ?></td>
    <td><?php echo $row['data_disp'] ?></td>
    <td><?php echo $row['n_targa'] ?></td>
    <td>
        <form action=<?php echo URL.'prenota/index'; ?>" method="post">
            <input type="hidden" name="id" value=<?php echo $row['id'] ?>>
            <button type="submit" class="button btn-primary">Prenota</button>
        </form>
    </td>
</tr>
<?php endforeach; ?>
```

Table 2 Input nascosto

Poi dal controller Prenota prendo l'id del posteggio e lo passo alla funzione che ricava i dati da poi mostrare nella pagina di prenotazione. Infine carico la pagina di prenotazione con i dati scaricati.

```
public static function index()
{
    $id = Validator::testInput($_POST['id']);
    PrenotaModel::getParcheggioInfo($id);
    ViewLoader::load( template: 'prenota/index', array('parcheggio'=>PrenotaModel::$parcheggio));
}
```

Table 3 Controller Prenota

Questa è la logica contenuta nella classe model.

```
class PrenotaModel
{
    public static $parcheggio;
    private static $statement;

    public static function getParcheggioInfo($id_parcheggio)
    {
        self::$statement = Database::get()->prepare(
            statement: "SELECT utente.nome, utente.cognome, utente.tel, parametri.costo, posteggio.data_disp,
                        posteggio.disponibilita, posteggio.n_targa
                        FROM utente, parametri, posteggio
                        WHERE posteggio.id = :id
                        AND utente.id posteggio = :id;
        ");
        self::$statement->bindParam( parameter: ':id', &variable: $id_parcheggio, data_type: \PDO::PARAM_INT );
        self::$statement->execute();
        self::$parcheggio = self::$statement->fetch( fetch_style: \PDO::FETCH_ASSOC );

        if(is_null(self::$parcheggio['n_targa']))
        {
            $row['n_targa'] = "Targa non fornita";
        }
        $inputDate = date_create(self::$parcheggio['data_disp']);
        $inputDateFormat = date_format($inputDate, format: 'd-m-Y');
        self::$parcheggio['data_disp'] = $inputDateFormat;
    }

    public static function prenota(){
    }
}
```

Table 4 Classe PrenotaModel

Nella variabile statement salvo il risultato ottenuto dalla query. In seguito applico delle modifiche ai campi n\_targa e data\_disp:

- n\_targa se è nullo lo modifco in “Targa non fornita”
- data\_disp lo formato nel formato che usiamo.

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

In linea.

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

Finire lo sviluppo del metodo per la prenotazione e iniziare il capitolo sull'implementazione della documentazione.

# Diario di lavoro

Luogo	Canobbio
Data	15.11.2019

## Lavori svolti

Durante la giornata di oggi ho avuto un colloquio con il docente formatore. Tutto ciò di cui abbiamo discusso è stato riportato nell'apposito file dentro la cartella Colloqui.

Oggi mi sono occupato della documentazione. In particolare ho svolto i capitoli 1.3, 2.1, 3.1.1, 3.1.2, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6, 3.3.7.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In linea.

## Programma di massima per la prossima giornata di lavoro

Aggiornare il gantt e iniziare ciò di cui abbiamo discusso con il docente formatore.

# Diario di lavoro

Luogo	Canobbio
Data	19.11.2019

## Lavori svolti

Durante la giornata di oggi ho iniziato a sviluppare il metodo di filtraggio dei posteggi. Ho svolto il filtraggio in base alla disponibilità.

Il risultato finale è il seguente:

The screenshot shows a web application for managing parking spaces. At the top, there is a navigation bar with links for 'Gestione Parcheggi', 'Home', 'Offerta', and 'Ricerca'. On the right side of the header is an 'Account' dropdown menu. Below the header, there is a map of a parking area with various spots numbered (e.g., 205, 206, 207, 208, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229). Overlaid on the map are two input fields: 'Inserisci le date da cercare:' with two empty text boxes for 'da' and 'al', and 'Inserisci la disponibilità:' with a dropdown menu set to 'Mattina'. Below these inputs is a table with the following data:

Disponibilità	Data disponibilità	Numero di targa	Prenota
Mattina	02-12-2019	TI-234567	<button>Prenota</button>
Mattina	16-11-2019	TI-234234	<button>Prenota</button>

Per riuscire a fare il tutto ho aggiunto il nuovo input per la disponibilità alla pagina dei risultati. L'input è lo stesso della pagina di offerta con una modifica per aggiungere lo stato di selected al valore selezionato.

```
<div class="col-6">
    <p class="text-white font-weight-bold">Inserisci la disponibilità:</p>
    <form method="post">
        <select class="browser-default custom-select w-50" name="select_disp" onchange="filterDisp(this.value)">
            <option value="Tutto il giorno" <?php echo ($selected == 'Tutto il giorno')?'selected':''?>>Tutto il giorno</option>
            <option value="Mattina" <?php echo ($selected == 'Mattina')?'selected':''?>>Mattina</option>
            <option value="Pomeriggio" <?php echo ($selected == 'Pomeriggio')?'selected':''?>>Pomeriggio</option>
        </select>
    </form>
</div>
```

In seguito ho creato il metodo di filtraggio nel rispettivo Model.

```
/**
 * Funzione che filtra i posteggi disponibili in base alla disponibilità scelta dall'utente.
 *
 * @param $disp Disponibilità selezionata nel filtro.
 */
public static function filterByDisp($disp)
{
    self::$statement = Database::get()->prepare( statement: "select * from posteggio where data_disp is not null and disponibilita = :disp");
    self::$statement->bindParam( parameter: ':disp', &variable: $disp, data_type: \PDO::PARAM_STR);
    self::$statement->execute();
    self::$parcheggi = self::$statement->fetchAll( fetch_style: \PDO::FETCH_ASSOC);

    foreach(self::$parcheggi as &$row)
    {
        if(is_null($row['n_targa']))
        {
            $row['n_targa'] = "";
        }
        $inputDate = date_create($row['data_disp']);
        $inputDateFormat = date_format($inputDate, format: 'd-m-Y');
        $row['data_disp'] = $inputDateFormat;
    }
}
```

In questo metodo faccio la stessa select di base che carica tutti i posteggi disponibili, ma gli aggiungo anche il valore disponibilità da controllare.

Successivamente ho modificato il codice presente nel controller per fare in modo di visualizzare il cambiamento dei dati una volta filtrati.

```
/**
 * Funzione che richiama il metodo per ricevere i posteggi offerti e poi reindirizza verso la pagina che li mostra.
 */
public function index()
{
    $selected = null;
    if(isset($_GET['filter_disp']))
    {
        $disp = Validator::testInput($_GET['filter_disp']);
        RicercaModel::filterByDisp($disp);
        $selected = $disp;
    }
    else {
        RicercaModel::getParcheggiDisponibili();
    }
    ViewLoader::load( template: 'ricerca/index', array('parcheggi'=>RicercaModel::$parcheggi, 'selected' => $selected));
}
```

Tutto ciò per funzionare ha bisogno della seguente funzione javascript che prende il valore selezionato nella select e lo invia tramite GET al controller.

```
function filterDisp(disp) {
    window.location.href = "/ricerca/index/?filter_disp=" + disp;
}
```

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

In linea.

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

Finire il filtraggio in base a un range di date e continuare i lavori assegnatimi dal docente.

# Diario di lavoro

Luogo	Canobbio
Data	21.11.2019

## Lavori svolti

Durante la giornata di oggi ho finito il filtraggio tramite un range di date e ho modificato i percorsi del progetto per fare in modo di averli uguali a quelli di Infomaniak. Ho inoltre aggiornato il Gantt.

Il risultato finale è il seguente:

The screenshot shows a web application for managing parking spaces. At the top, there is a navigation bar with links for 'Gestione Parcheggi', 'Home', 'Offerta', and 'Ricerca'. Below the navigation, there is a search form with fields for 'Inserisci le date da cercare:' (From date) and 'Inserisci la disponibilità:' (Availability), along with a 'Cerca' (Search) button. To the right of the search form is a dropdown menu set to 'Tutto il giorno' (All day). The main area features a grayscale map of a parking lot with various parking spots numbered (e.g., 205, 206, 207, 208, 213, 214, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230). Overlaid on the map is a table of available parking spaces:

Disponibilità	Data disponibilità	Numero di targa	Prenota
Mattina	02-12-2019	TI-234567	Prenota
Mattina	16-11-2019	TI-234234	Prenota
Tutto il giorno	20-11-2019	GR-12345	Prenota
Tutto il giorno	23-11-2019		Prenota
Proprietario	Disponibilità	Data disponibilità	Prenota

Per riuscire a fare il tutto ho aggiunto i nuovi datepicker e un pulsante per eseguire il filtraggio alla pagina dei risultati.

```
<div class="col-6">
    <p class="text-white font-weight-bold">Inserisci le date da cercare:</p>
    <div class="input-group justify-content-center">
        <p class="text-white font-weight-bold ml-3 mr-3">Dal </p>
        <input type="text" id="startDate" name="startDate" class="datepicker">
        <p class="text-white font-weight-bold ml-3 mr-3"> al </p>
        <input type="text" id="endDate" name="endDate" class="datepicker">
        <input type="submit" value="Cerca" class="button btn-primary ml-3" onclick="filterDate()">
    </div>
```

In seguito ho creato il metodo di filtraggio nel rispettivo Model.

```
/***
 * Funzione che filtra i posteggi disponibili in base alle date scelte dall'utente.
 */
public static function filterByDate($startDate, $endDate)
{
    $dateFormat = date_format(date_create($startDate), format: "Y-m-d H:i:s");
    $endDateFormat = date_format(date_create($endDate), format: "Y-m-d H:i:s");

    if($dateFormat > $endDateFormat)
    {
        $_SESSION['minDateError'] = 'Errore: la data iniziale è più grande di quella finale';
    }
    else
    {
        unset($_SESSION['minDateError']);
    }

    self::$statement = Database::get()->prepare(statement: "select * from parcheggio where data_disp between :start_date and :end_date");
    self::$statement->bindParam(parameter: ':start_date', &variable: $dateFormat, data_type: \PDO::PARAM_STR);
    self::$statement->bindParam(parameter: ':end_date', &variable: $endDateFormat, data_type: \PDO::PARAM_STR);
    self::$statement->execute();

    self::$parcheggi = self::$statement->fetchAll(fetch_style: \PDO::FETCH_ASSOC);

    self::formatCarAndDate();
}
```

In questo metodo faccio la stessa select di base che carica tutti i posteggi disponibili, ma gli aggiungo anche il valore data\_disp da controllare.

Ho modificato il codice presente nel controller per fare in modo di visualizzare il cambiamento dei dati una volta filtrati.

```
/***
 * Funzione che richiama il metodo per ricevere i posteggi offerti e poi reindirizza verso la pagina che li mostra.
 */
public function index()
{
    $selected = null;
    if(isset($_GET['filter_disp']))
    {
        $disp = Validator::testInput($_GET['filter_disp']);
        RicercaModel::filterByDisp($disp);
        $selected = $disp;
    }
    elseif (isset($_GET['startDate']) && isset($_GET['endDate']))
    {
        $startDate = $_GET['startDate'];
        $endDate = $_GET['endDate'];

        RicercaModel::filterByDate($startDate, $endDate);
    }
    else {
        RicercaModel::getParcheggiDisponibili();
    }
    ViewLoader::load(template: 'ricerca/index', array('parcheggi'=>RicercaModel::$parcheggi, 'selected' => $selected));
}
```

Tutto ciò per funzionare ha bisogno della seguente funzione javascript che prende i valori selezionato nei datepicker e li invia tramite GET al controller.

```
function filterDate() {  
    var startDate = document.getElementById('startDate').value;  
    var endDate = document.getElementById('endDate').value;  
    window.location.href = "/gestioneparcheggi2019/ricerca/index/?startDate=" + startDate + "&endDate=" + endDate;  
}
```

#### Problemi riscontrati e soluzioni adottate

-

#### Punto della situazione rispetto alla pianificazione

In linea.

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

Caricare il sito online e fare il metodo per la prenotazione.

# Diario di lavoro

Luogo	Canobbio
Data	22.11.2019

## Lavori svolti

Durante la giornata di oggi ho avuto il colloquio con il docente formatore. I dettagli sono disponibili nella cartella Colloqui. Oggi ho finito il meccanismo di prenotazione di un parcheggio. Ora è possibile prenotare un parcheggio e una volta fatto quest'ultimo verrà rimosso dalla lista dei parcheggi disponibili.

Nel model per la prenotazione ho aggiunto del codice con il quale salvo l'id del parcheggio nella sessione perchè mi servirà in seguito.

```
public static function getParcheggioInfo($id_parcheggio)
{
    $SESSION['id_posteggio_prenotato'] = $id_parcheggio;
```

Ho creato due nuove funzioni: prenota() e updateParcheggio(). La prima serve per aggiungere una prenotazione mentre la seconda aggiorna lo stato del parcheggio prenotato.

```
/** 
 * Funzione che crea una prenotazione.
 */
public static function prenota()
{
    self::$statement = Database::get()->prepare("insert into prenotazione
        (richiamo, data_prenotazione, id_utente, id_posteggio)
        values
        (false, current_timestamp, :id_utente, :id_posteggio);
    ");

    self::$statement->bindParam(':id_utente', $_SESSION['user_id'], \PDO::PARAM_INT);
    self::$statement->bindParam(':id_posteggio', $SESSION['id_posteggio_prenotato'], \PDO::PARAM_INT);

    try
    {
        if (Auth::isAuthenticated())
        {
            self::$statement->execute();
            self::updateParcheggio();
            unset($_SESSION['id_posteggio_prenotato']);
            ViewLoader::load('home/index', array('prenotazioneOK'=>"Prenotazione avvenuta"));
        }
        else {
            ViewLoader::load("register/index", array('prenotazioneNO'=>"Devi prima registrarti"));
        }
    } catch (PDOException $e)
    {
        ViewLoader::load('prenotazione/index', array('parcheggio'=>self::$parcheggio,
        'prenotazioneNO'=>"Prenotazione fallita"));
    }
}
```

Questa è la funzione prenota. Se l'utente che esegue la prenotazione è registrato eseguo la query, aggiorno lo stato del parcheggio, unsetto la variabile nella sessione con l'id e poi carico la pagina home nella quale mostro una notifica di successo. Nella pagina home ho aggiunto il seguente codice per far visualizzare la notifica:

```
echo isset($prenotazioneOK)? "<script> $.notify(\"$prenotazioneOK.\", \"success\")</script>": "";
```

```
/**  
 * Funzione che aggiorna lo stato di un parcheggio prenotato.  
 */  
public static function updateParcheggio()  
{  
    self::$statement = Database::get()->prepare("update posteggio  
        set disponibilita=null, data_disp=null, n_targa=null  
        where id=:id;  
    ");  
    self::$statement->bindParam(':id', $_SESSION['id_posteggio_prenotato'], \PDO::PARAM_INT);  
    self::$statement->execute();  
}
```

Questa è invece la funzione updateParcheggio che setta tutti i campi di un parcheggio a null affichè il suo stato di “messo in offerta” venga annullato. Se l’utente invece prova a eseguire una prenotazione non essendo registrato, viene reindirizzato alla pagina di registrazione e viene mostrata una notifica che lo spiega. Il codice è il seguente:

```
echo isset($prenotazioneNO)?<script> $.notify(\"".$prenotazioneNO."\","  
\\"error\")</script>": "";
```

Quando si eseguiva un’offerta correttamente, veniva caricata una pagina bianca e l’utente non veniva notificato dell’effettiva messa in offerta del posteggio. Ho fatto una modifica al codice e ora si viene reindirizzati nella pagina home e viene visualizzata una notifica di successo. Il codice è il seguente:

```
echo isset($offertaOK)?<script> $.notify(\"".$offertaOK."\","  
\\"success\")</script>": "";
```

Nel model per l’offerta ho aggiunto invece il seguente codice che carica la pagina home passandogli come parametro il valore da inserire nella notifica:

```
ViewLoader::load('home/index', array('offertaOK'=>"Offerta andata a buon fine"));
```

Nella view home ho inoltre aggiunto il meta contenente la versione del sito.

```
<meta name="versione" content="00.06.00-22.11.2019">
```

### Problemi riscontrati e soluzioni adottate

-

### Punto della situazione rispetto alla pianificazione

In ritardo.

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

Continuare la documentazione e svolgere i compiti definiti nel rapporto del colloquio odierno.

# Diario di lavoro

Luogo	Canobbio
Data	26.11.2019

## Lavori svolti

Durante la giornata di oggi mi sono portato avanti con la documentazione. Ho documentato tutti i controller che ho creato fin'ora.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Finire la documentazione della parte di implementazione svolta fin'ora e continuare con i lavori concordati con il docente formatore.

# Diario di lavoro

Luogo	Canobbio
Data	28.11.2019

## Lavori svolti

Durante la giornata di oggi mi sono portato avanti con la documentazione. Ho documentato un Model. Ho sistemato l'errore sulla pagina della riservazione del quale ho discusso con il docente formatore durante un colloquio avvenuto nella pausa sul mezzogiorno. Tutti i dettagli sul colloquio sono disponibili nella cartella Colloqui.

Ho scaricato PHPMailer per poter inviare una mail alla registrazione di un utente e alla sua prenotazione. Dovrò documentarmi sul uso funzionamento per poi poterlo integrare appieno nel progetto.

## Problemi riscontrati e soluzioni adottate

Oggi è arrivato il docente Mussi con i ragazzi del Promtec. Uno di loro mi è stato assegnato e quindi gli ho spiegato il progetto che sto svolgendo. Durante la demo mi sono accorto di un errore nella pagina di offerta dei parcheggi. Se si inseriva una data non valida mostrava l'errore della data non valida, ma eseguiva comunque il codice dell'inserimento dell'offerta. Ho risolto mettendo tutto il codice per l'offerta in un else.

```
if(isset($_SESSION['dateError']) || isset($_SESSION['carPlateError']))  
{  
    ViewLoader::load('offerta/index');  
  
    unset($_SESSION['dateError']);  
    unset($_SESSION['carPlateError']);  
}  
else  
{  
    if(Users::hasParcheggio())  
    {  
        $inputDate = date_create($self::$selectedDate);  
        $inputDateFormat = date_format($inputDate, "Y-m-d H:i:s");  
    }  
}
```

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Integrare PHPMailer nel progetto e svolgere i compiti discussi nei colloqui.

# Diario di lavoro

Luogo	Canobbio
Data	29.11.2019

## Lavori svolti

Dirante la giornata di oggi ho svolto un colloquio con il docente formatore. I dettagli sono disponibili nella sezione Colloqui su GitHub.

Ho integrato PHPMailer nel progetto. Quando un utente si registra al sito riceve una mail con un link tramite il quale poter attivare l'account. Ho scaricato i file necessari di PHPMailer dalla repo su GitHub e li ho caricati nel progetto nella cartella Libs/PHPMailer. Ho aggiunto nel file di config la directory per eseguire l'autoload dei file.

```
$autoload_directories = array(
    "application/controllers/",
    "application/libs/",
    "application/libs/phpmailer/",
    "application/models/"
);
```

Ho creato la classe model MailModel per la gestione delle mail. Ha una proprietà mail privata che conterrà la mail da inviare.

```
public static function build()
{
    self::$mail = new PHPMailer(true);
    self::$mail->isSMTP();
    self::$mail->Host = 'smtp.gmail.com';
    self::$mail->Port = 587;
    self::$mail->SMTPAuth = true;
    self::$mail->Username = 'gestioneparcheggi.samt@gmail.com';
    self::$mail->Password = 'Parcheggi_Admin_2019';
    self::$mail->setFrom('gestioneparcheggi.samt@gmail.com', 'Gestione parcheggi');
}
```

Il metodo build() crea una nuova mail con le informazioni di base richieste per ogni mail.

```
public static function newUserMail($newUserMail, $nome, $cognome){
    self::build();
    self::$mail->addAddress($newUserMail, $nome . " " . $cognome);
    self::$mail->Subject = "Registrazione effettuata";
    self::$mail->Body = "Il tuo account è stato creato con successo!<br><br>
                            Per poter essere in grado di usare il tuo account clicca sul
                            seguente link per attivarlo:<br>
                            https://samtinfo.ch/gestioneparcheggi2019/register/verify/?mail=". $newUserMail . "&nome=
                            ". $nome .
                            "&cognome=". $cognome;
    self::$mail->send();
}
```

Il metodo newUserMail(\$newUserMail, \$nome, \$cognome) serve per formattare la mail da inviare quando un utente si registra al sito e deve attivare l'account. Riceve come parametri la mail, il nome e il cognome dell'utente. Ho lasciato il nome e il cognome per eseguire il testing della classe, ma durante le prossime ore aggiungerò al campo mail del database

l'attributo UNIQUE per poter passare al metodo solo la mail dell'utente.

Nel controller Registration ho aggiunto il metodo verify() che richiama il metodo per la verifica creato nel model RegistrationModel.

```
/**  
 * Funzione che richiama il metodo per la verifica dell'utente.  
 */  
public function verify()  
{  
    if(isset($_GET['mail']) && isset($_GET['nome']) && !empty($_GET['cognome'])){  
  
        $mail = Validator::testInput($_GET['mail']);  
        $nome = Validator::validateCharAndSpace($_GET['nome']);  
        $cognome = Validator::validateCharAndSpace($_GET['cognome']);  
  
        RegisterModel::verify($mail, $nome, $cognome);  
    }else{  
  
        ViewLoader::load('home/index', array('activationNO'=>"Usa il link che ti è  
        stato inviato per mail!"));  
    }  
}
```

Nel controller prendo i dati inviati nell'header tramite una richiesta get e li passo alla funzione verify() nel model.

```
public static function verify($mail, $nome, $cognome)  
{  
    self::$statement = Database::get()->prepare("select mail, nome, cognome  
        from utente where mail=:mail and nome=:nome and cognome=:cognome;  
    ");  
  
    self::$statement->bindParam(':mail', $mail, PDO::PARAM_STR);  
    self::$statement->bindParam(':nome', $nome, PDO::PARAM_STR);  
    self::$statement->bindParam(':cognome', $cognome, PDO::PARAM_STR);  
  
    self::$statement->execute();  
    $result = self::$statement->fetch(\PDO::FETCH_ASSOC);  
  
    if($result > 0){  
        self::$statement = Database::get()->prepare("update utente set attivo=true  
            where mail=:mail and nome=:nome and cognome=:cognome and  
            attivo=false  
        ");  
  
        ViewLoader::load('home/index', array('activationOK'=>"Il tuo account è stato  
        attivato con successo!"));  
    }else{  
        ViewLoader::load('home/index', array('activationNO'=>"L'URL è invalido o il  
        tuo account è già attivo!"));  
    }  
}
```

In questa funzione preparo la query che prende i valori dal database per confrontarli con quelli passati dal browser. Se c'è una corrispondenza aggiorno lo stato della proprietà attivo a true e carico la view home con un messaggio di successo. Se invece non c'è una corrispondenza significa che l'url non è valido o che l'utente è già stato attivato.

**Problemi riscontrati e soluzioni adottate**

L'invio della mail funziona, ma il link da premere per l'attivazione dell'account non va. Devo ancora capire come mai.

**Punto della situazione rispetto alla pianificazione**

In ritardo.

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

Risolvere il problema con il link di attivazione, integrare l'invio della mail quando si effettua una prenotazione e svolgere i compiti discussi durante il colloquio.

# Diario di lavoro

Luogo	Canobbio
Data	03.12.2019

## Lavori svolti

Durante la giornata di oggi ho risolto il problema del link di verifica. La soluzione è descritta nel capitolo successivo.

Ho integrato l'invio di una mail quando si esegue una prenotazione. L'utente riceverà anche un pdf con un resoconto di quest'ultima. La formattazione del pdf è ancora da discutere con il docente formatore. Questa è solo una bozza.

## Dati della riservazione

ID prenotazione	Data prenotazione	Disponibilità	Data disponibilità
55	2019-12-03 16:18:44	Tutto il giorno	2020-01-02 00:00:00

*Table 1 Bozza della ricevuta di prenotazione*

Per creare il pdf ho usato la libreria fpdf che con il docente Sartori abbiamo già utilizzato.

## Problemi riscontrati e soluzioni adottate

Il link per la verifica della mail funzionava, ma facevo il redirect al sito sull'hosting che però non era aggiornato. Ho modificato il percorso puntando al server sulla mia macchina.

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Svolgere i compiti discussi durante l'ultimo colloquio.

# Diario di lavoro

---

Luogo	Canobbio
Data	05.12.2019

## Lavori svolti

**Oggi non abbiamo avuto lezioni perché eravamo in visita alla SUP di Friborgo.**

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Svolgere i compiti discussi durante l'ultimo colloquio.

# Diario di lavoro

Luogo	Canobbio
Data	06.12.2019

## Lavori svolti

Durante la giornata di oggi ho avuto un colloquio con il docente formatore. I dettagli del colloquio sono disponibili nella cartella Diari su GitHub.

Alla select per il filtraggio in base alla disponibilità ho aggiunto, come discusso assieme al docente formatore, un'opzione in più per poter permettere di cercare tutti i posteggi disponibili. Questo perchè se si eseguiva un filtraggio per poter riavere tutti i posteggi disponibili bisognava ricaricare tutta la pagina. Il codice è il seguente:

```
if($disp == "Tutto")
{
    self::$statement = Database::get()->prepare("select * from posteggio where
data_disp is not null");
}
else
{
    self::$statement = Database::get()->prepare("select * from posteggio where
data_disp is not null and disponibilita = :disp");
    self::$statement->bindParam(':disp', $disp, \PDO::PARAM_STR);
}
```

Se è selezionata l'opzione “Tutto” interrogo il database senza nessuna restrizione. Se invece il valore è diverso faccio la query in base al filtro scelto.

Ho aggiunto i commenti al codice nuovo. Ho creato lo script per l'aggiunta di prenotazioni.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Svolgere i compiti discussi durante l'ultimo colloquio.

# Diario di lavoro

Luogo	Canobbio
Data	10.12.2019

## Lavori svolti

Durante la giornata di oggi ho modificato alcuni parametri per l'aggiunta di dati al database. Ho modificato le varie date di prenotazione dei posteggi affinchè non fossero tutte uguali.

In seguito ho modificato la struttura per il filtraggio dei dati. Pensavo fosse qualcosa di veloce, invece ho dovuto cambiare il modo in cui prendo i dati dalla view e li invio al controller. Il risultato finale è il seguente:

Disponibilità	Data disponibilità	Numero di targa	Prenota
Mattina	10-12-2019	TI-147566	<button>Prenota</button>
Mattina	10-12-2019	TH120677	<button>Prenota</button>
Mattina	10-12-2019	TI-259650	<button>Prenota</button>
Mattina	10-12-2019	TI-295424	<button>Prenota</button>
Mattina	10-12-2019	TI-129780	<button>Prenota</button>
Mattina	10-12-2019	TI-135907	<button>Prenota</button>

Table 1 Nuovo layout per il filtraggio dei dati

Nella view ho modificato il codice relativo all'html. Ho spostato il tutto dentro a un form che con un metodo in post richiama il metodo index del controller Research.

```
<form method="post" action="php echo URL . 'research/index'; ?&gt;&gt;
  &lt;div class="col"&gt;

    &lt;p class="text-white font-weight-bold"&gt;Inserisci le date da cercare:&lt;/p&gt;
    &lt;div class="input-group justify-content-center"&gt;
      &lt;input type="text" id="startDate" name="startDate" class="datepicker"&gt;
      &lt;p class="text-white font-weight-bold ml-3 mr-3"&gt; al &lt;/p&gt;
    &lt;/div&gt;
  &lt;/div&gt;
&lt;/form&gt;</pre
```

```

<input type="text" id="endDate" name="endDate" class="datepicker">

</div>

</div>

<div class="col">

    <p class="text-white font-weight-bold"><br>Inserisci la disponibilità:</p>

    <div class="input-group justify-content-center">

        <select class="browser-default custom-select w-50" name="filter_disp"
id="disp">
            <option value="Tutto" <?php echo ($selected ==
'Tutto')?'selected':''?>>Tutto</option>
            <option value="Tutto il giorno" <?php echo ($selected == 'Tutto il
giorno')?'selected':''?>>Tutto il giorno</option>
            <option value="Mattina" <?php echo ($selected ==
'Mattina')?'selected':''?>>Mattina</option>
            <option value="Pomeriggio" <?php echo ($selected ==
'Pomeriggio')?'selected':''?>>Pomeriggio</option>
        </select>

    </div>

    <br>
    <input type="submit" value="Cerca" class="button btn-primary ml-3">

</div>

</form>

```

Ho cancellato le funzioni JavaScript con le quali prendevo i dati dai due datepicker e dalla select. Ora tramite il form prendo i dati inviati con un metodo post e gestisco la scelta del metodo di filtraggio nel controller.

```

public function index()
{
    $selected = null;
    if (isset($_POST['startDate']) && !empty($_POST['startDate']) &&
isset($_POST['endDate']) && !empty($_POST['endDate']) && isset($_POST['filter_disp']))
    {
        $startDate = $_POST['startDate'];
        $endDate = $_POST['endDate'];
        $disp = Validator::testInput($_POST['filter_disp']);
        $selected = $disp;

        ResearchModel::filterAll($startDate, $endDate, $disp);
    }
    elseif (isset($_POST['filter_disp']))
    {
        $disp = Validator::testInput($_POST['filter_disp']);
        $selected = $disp;

        ResearchModel::filterByDisp($disp);
    }
    else
    {

```

```
    ResearchModel::getParcheggiDisponibili();
}
ViewLoader::load('ricerca/index', array('parcheggi'=>ResearchModel::$parcheggi,
'selected' => $selected));
}
```

Nel primo if controllo che la le date di inizio e fine ricerca siano settate ma non vuote e se la select è settata. Il controllo sulla select effettivamente potrebbe essere tolto perchè di default sarà settato sul valore “Tutti”. Il secondo if viene richiamato quando le date di inizio e fine sono vuote. Questo significa che l’utente, in teoria, vuole fare una ricerca solo in base alla disponibilità quindi la valore nella select. In tutti gli altri casi carico tutti i posteggi disponibili.

In ResearchModel ho modificato il metodo filterByDate in filterAll facendo in modo di effettuare una richiesta al database con le due date settate e la disponibilità.

Nel MailModel ho aggiunto la cifrature in UTF-8 perchè alcuni caratteri nelle mail inviate non venivano interpretati correttamente:

```
public static function build()
{
    self::$mail = new PHPMailer(true);
    self::$mail->CharSet = 'UTF-8';
    self::$mail->isSMTP();
```

Ho caricato la versione del sito aggiornata sull’hosting. Ho testato quanto fatto finora e funziona.

### Problemi riscontrati e soluzioni adottate

### Punto della situazione rispetto alla pianificazione

In ritardo.

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

Continuare possibilmente la documentazione. Aggiornare la parte dei controller e finire la documentazione dei models. Finire il gantt intermedio. Iniziare l’abstract.

# Diario di lavoro

Luogo	Canobbio
Data	12.12.2019

## Lavori svolti

Durante la giornata di oggi mi sono portato avanti ocn la stesura della documentazione. Ho aggiornato la parte del controller Register aggiungendo il metodo per la verifica della mail.

Nella parte dei Model ho fatto il ResearchModel, ReserveModel, RegisterModel e OfferModel.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Continuare possibilmente la documentazione. Aggiornare la parte dei models e il gantt.

# Diario di lavoro

Luogo	Canobbio
Data	13.12.2019

## Lavori svolti

Durante la giornata di ho finito la documentazione dei models. L'unico model che non ho documentato è quello per la creazione dei pdf perchè è ancora un work in progress. Lo documenterò una volta finito.

Ho aggiornato il gantt intermedio.

Ho stampato tutti i diari fatti finora in modo da portarmi avanti con la consegna del progetto.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

In ritardo.

## Programma di massima per la prossima giornata di lavoro

Finire l'impostazione del pdf della prenotazione e documentarlo. Finire gli altri capitoli della documentazione.