

UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione ed

Elettrica e Matematica Applicata (DIEM)

Corso di Software Engineering



Progettazione e sviluppo di una applicazione web per la
gestione delle attività di manutenzione con la
metodologia SCRUM

Team 2

Alessandra Vitolo 06227701496

Lorenzo Laudato 06227701563

Martina Lamberti 06227701476

Sonia Mola 06227701562

Anno accademico 2020/2021

Indice

Introduzione

Contesto applicativo	4
1 Metodologia SCRUM	5
1.1 User Stories	6
1.2 Product Backlog	7
1.3 Sprint	8
1.4 Trello	12
2 Progettazione	13
2.1 Architettura software	13
2.2 Diagramma UML delle classi	16
2.3 Pattern applicati	17
2.4 Tecnologie utilizzate	18
2.5 Database	19
3 Sviluppo	22
3.1 Struttura dell'app	22
3.2 Navigazione dell'app	28
4 Testing	39
4.1 Testing della GUI	39
4.2 Testing unitario con PHPUnit	41

Introduzione

Contesto applicativo

Il termine manutenzione si può riferire a compiti svolti in settori molto diversi fra loro e in tutti i tipi di ambienti di lavoro. È un'attività che comprende tutte le azioni tecniche, amministrative e gestionali, eseguite durante il ciclo di vita di un elemento destinate a preservarlo o a riportarlo in uno stato in cui possa eseguire la funzione richiesta.

L'obiettivo di questo progetto è la progettazione e lo sviluppo di una *web application* finalizzata alla gestione delle attività di manutenzione con l'utilizzo della metodologia SCRUM.

Capitolo 1

Metodologia SCRUM

Il progetto è stato eseguito con SCRUM, un framework capace di sviluppare e sostenere prodotti complessi. Tutto il lavoro che c'è dietro lo sviluppo dell'applicazione ha fedelmente preso in considerazione i tre pilastri propri di SCRUM: *trasparenza*, *ispezione* e *adattamento*.

Durante tutte le fasi che hanno interessato lo sviluppo dell'app ogni componente del team ha avuto piena coscienza dell'obiettivo fissato (trasparenza verticale) e di quali fossero i compiti affidati agli altri membri (trasparenza orizzontale). *Trasparenza*, intesa anche nel senso che il lavoro e le relative performance sono state visibili a tutti a qualsiasi livello organizzativo.

Relativamente all'*ispezione*, il team ha ispezionato di frequente il prodotto mentre lo si stava sviluppando.

L'*adattamento* è la conseguenza dell'ispezione e significa che, al posto di seguire un piano preordinato, il team di sviluppo ripianifica in base ai risultati dell'ispezione per apportare il maggior valore al cliente finale orientato al miglioramento continuo delle proprie performance. Per tale motivo, gli artefatti previsti dalla metodologia SCRUM sono stati modificati di frequente.

Per far sì che questi pilastri vengano attuati, SCRUM prevede che tutto il personale coinvolto nello sviluppo condivida valori come "concentrazione", "coraggio", "apertura", "impegno" e "rispetto".

Per mantenere alta la *concentrazione*, il team non si è sottoposto a ritmi stressanti e si è concentrato su pochi task alla volta.

Come tecnica di sviluppo software si è scelto di utilizzare il pair programming. Affinché il team nella sua totalità restasse sempre aggiornato, i ruoli sono stati scambiati vicendevolmente. Avere il supporto e riconoscere negli altri membri delle risorse dà il *coraggio* per affrontare compiti impegnativi.

Per quanto riguarda l'*apertura*, è importante che tutti si aprano agli altri membri del team, spiegando ciò che si sta facendo e gli eventuali problemi che si incontrano. Ogni componente del team, in corso d'opera, ha espresso le sue preoccupazioni in maniera tale che potessero essere affrontate e risolte in gruppo.

I valori come *impegno* e *rispetto* sono stati i cardini fondamentali per conseguire gli obiettivi prefissati dal team.

Artefatti

Gli artefatti sono tre: *Product Backlog*, *Sprint Backlog* e *Incremento*. Sono pensati per aumentare la trasparenza delle informazioni principali, così che tutto il Team ne sia al corrente.

In Scrum il prodotto finale viene realizzato in modo incrementale, lavorando step by step alle varie funzionalità: queste sono quindi ordinate secondo la priorità con la quale si desidera che siano implementate.

1.1 User Stories

USER STORY ID	PRIORITY	AS A <type of user>	I WANT TO <perform some task>	SO IT IS POSSIBLE TO <achieve some goal>
1	Low	System Administrator	create, view, modify or delete system users, assign them username, password, and a specific role (planner, maintainer)	manage users who will have access to the system
2	High	System Administrator	know each maintenance activity	associate to each maintainer the procedure that he can perform
3	High	Planner	select a specific activity to know the intervention information	verify the skills and resources needed for the intervention
4	Medium	System Administrator	have a Standard Maintenance Procedure (SMP) file in PDF format	associate it for each Maintenance Procedure
5	Medium	System Administrator	view or modify workspace notes	manage workspace notes and associate them to a specific entity or a set of entity
6	Low	System Administrator	record all access to the application	track any users' actions
7	High	Planner	know informations about activities organized by week	manage (view or modify) maintenance activities.
8	High	System Administrator	know the skills of each maintainer	allow to assign specific competencies for each maintainer
9	High	Planner	see competencies required and the list of Maintainers	assign the scheduled activity to a specific Maintainer, according to his availability.
10	Medium	Planner	see maintainer name, maintainer availability percentage, maintainer competencies compliance, availability (in minutes for each hour of his workday)	select among the days of the week that the Maintainer has availability
11	High	Planner	select the slot of availability time	assign the maintenance activity and program the activity
12	Medium	Planner	send a notification to the selected Maintainer profile with a copy by e-mail to the Production manager	inform the maintainer about his new assigned activity
13	High	Planner	add intervention description, estimated intervention time, competencies required to perform the intervention when a EWO activity is selected	manage a EWO
14	Low	Planner	receive a notification from the selected Maintainer profile	know that his maintenance activity is finished
15	Medium	Planner	see the list of assigned tickets (EWO)	know about the tickets state
16	High	Repository Manager	manage maintenance typologies as Electrical, electronic, hydraulic, mechanical	distinguish different types of activity
17	High	Planner	define the time and send the communication to the maintainer and production manager	assign the non-scheduled activity to a specific maintainer

Le User Stories sono brevi e semplici descrizioni di una funzionalità raccontata dal punto di vista dell'utente o del cliente del sistema. Nella figura sovrastante sono presenti tutte le User Stories individuate dal team sulla base del documento SRS (SOFTWARE REQUIREMENT SPECIFICATION) fornito nelle fasi embrionali del progetto.

1.2 Product Backlog

Il Product Backlog è la lista ordinata di tutti gli elementi che verranno implementati dal team. Elenca caratteristiche, funzioni, requisiti, miglioramenti e correzioni che costituiscono le modifiche da apportare alle versioni future del prodotto. Siccome i progetti Scrum accolgono volentieri il cambiamento, gli artefatti non sono fissi ma evolvono nel tempo. Il team ha effettuato non poche modifiche: man mano che il prodotto venisse costruito, gli artefatti crescevano di complessità e dovevano essere adattati ai cambiamenti. Attraverso l'uso di brevi Sprint iterativi, si prende coscienza dei *feedback* ottenuti nella precedente release. Ciò consente al cliente di interagire abitualmente con i membri dello Scrum Team, di visionare i deliverable non appena sono pronti e di cambiare i requisiti prima dell'inizio dello Sprint successivo, se necessario.

TASK ID	TASK NAME	SPRINT #	ASSIGNED TO	START	FINISH	STORY	SPRINT READY	PRIORITY	STATUS	STORY POINTS	ASSIGNED TO SPRINT
Sprint 1											
1	User Story #7	" "	Alessandra, Lorenzo, Sonia, Martina	23/11/2020	01/12/2020	Yes	Yes	High	Complete	8	Yes
2	User Story #16	" "	Alessandra, Lorenzo, Sonia, Martina	24/11/2020	24/11/2020	Yes	Yes	High	Complete	2	Yes
3	User Story #3	" "	Alessandra, Lorenzo, Sonia, Martina	24/11/2020	03/12/2020	Yes	Yes	High	Complete	8	Yes
4	User Story #4	" "	Alessandra, Lorenzo, Sonia, Martina	24/11/2020	26/11/2020	Yes	Yes	Medium	Complete	5	Yes
5	User Story #5	" "	Alessandra, Lorenzo, Sonia, Martina	24/11/2020	29/11/2020	Yes	Yes	Medium	Complete	2	Yes
Sprint 2											
6	User Story #9	" "	Alessandra, Lorenzo, Sonia, Martina	04/12/2020	07/12/2020	Yes	Yes	High	Complete	8	Yes
7	User Story #6	" "	Alessandra, Lorenzo, Sonia, Martina	11/12/2020	12/12/2020	Yes	Yes	High	Complete	5	Yes
8	User Story #10	" "	Alessandra, Lorenzo, Sonia, Martina	04/12/2020	13/12/2020	Yes	Yes	Medium	Complete	7	Yes
9	User Story #2	" "	Alessandra, Lorenzo, Sonia, Martina	11/12/2020	12/12/2020	Yes	Yes	High	Complete	5	Yes
10	User Story #11	" "	Alessandra, Lorenzo, Sonia, Martina	07/12/2020	10/12/2020	Yes	Yes	High	Complete	3	Yes
11	User Story #12	" "	Alessandra, Lorenzo, Sonia, Martina	11/12/2020	13/12/2020	Yes	Yes	Medium	Complete	8	Yes
12	User Story #1	" "	Alessandra, Lorenzo, Sonia, Martina	06/12/2020	11/12/2020	Yes	Yes	Low	Complete	5	No
13	User Story #13	" "	Alessandra, Lorenzo, Sonia, Martina	10/12/2020	13/12/2020	Yes	Yes	High	Complete	8	No

Il team ha concepito la suddivisione delle user stories in due Sprint, della durata di 10 giorni ciascuno. In rosso sono evidenziate le user stories che inizialmente non erano state prese in considerazione per il secondo Sprint: il team ha preso consapevolezza del fatto che le capacità acquisite hanno consentito di viaggiare più velocemente rispetto al primo Sprint.

1.3 Sprint

Per semplificare la lavorazione, ogni elemento del Product Backlog è stato scomposto in sotto-attività, anche definite task. Lo sviluppo del prodotto viene effettuato grazie alla sequenza dei task che lo compongono: mediante questa suddivisione è ancora più evidente il principio del pair programming. Il team ha deciso di abbracciare questa tecnica, assegnando ad ogni coppia uno specifico task.

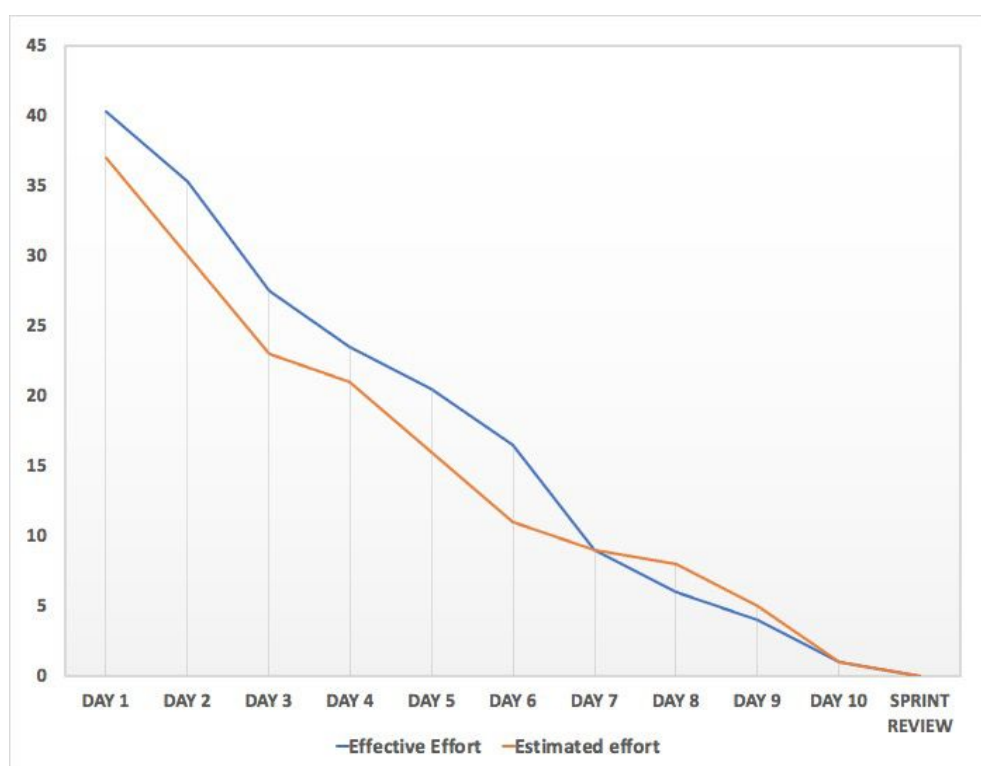
Lo Sprint Backlog è un ulteriore artefatto caratterizzato dall'insieme degli elementi del Product Backlog che sono stati selezionati per essere completati in una iterazione. Mediante questo artefatto, il team ha preso visione di quanto doveva essere prodotto alla fine dello Sprint per raggiungere l'obiettivo dell'iterazione, in gergo Sprint Goal. Anche questo artefatto è in continua evoluzione e tiene traccia del lavoro svolto giornalmente, delle attività completate e di quanto rimane da fare. Il team, in corso d'opera ha ritenuto opportuno aggiungere, eliminare o modificare task.

Durante lo Sprint Planning, che avviene all'inizio di ogni Sprint, il team ha concordato l'obiettivo dello Sprint: predisporre tutto il lato server necessario e comporre il database allo scopo di realizzare i primi due mockup presenti nel documento SRS.

Di seguito è riportato un estratto dello Sprint Backlog relativo al primo Sprint.

BACKLOG TASK & ID	STORY POINTS	ASSIGNED TO	STATUS	ORIGINAL EFFECTIVE	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	DAY 8	DAY 9	DAY 10
User Story #7	8	Alessandra, Lorenzo, Sonia, Martina												
Create the DB		Alessandra, Lorenzo, Sonia, Martina		9	5	4	0	0	0	0	0	0	0	0
Create the table in DB for the activities		Lorenzo, Martina		0,25	0	0,25	0	0	0	0	0	0	0	0
Create the web page (GUI)		Alessandra, Sonia		3	0	0	3	0	0	0	0	0	0	0
Implement web page's logic		Lorenzo, Martina		4	0	0	1	3	0	0	0	0	0	0
Testing phase		Alessandra, Sonia		3	0	0	0	0	0	0	0	2	1	0
User Story #16	2	Lorenzo, Martina												
Update the DB associating a specific type to each activity		Lorenzo, Martina		0,5	0	0,5	0	0	0	0	0	0	0	0
User Story #3	8	Alessandra, Lorenzo, Sonia, Martina												
Update the DB associating the information to each activity		Alessandra, Sonia		0,5	0	0,5	0	0	0	0	0	0	0	0
Create the web page (GUI)		Lorenzo, Martina		5	0	0	0	0	4	1	0	0	0	0
Implement web page's logic		Alessandra, Sonia		3	0	0	0	0	0	3	0	0	0	0
Testing phase		Lorenzo, Martina		3	0	0	0	0	0	0	0	0	2	1
User Story #4	5	Alessandra, Lorenzo, Sonia, Martina												
Update the database inserting the SMP file in PDF format of the selected activity		Alessandra, Martina		2	0	2	0	0	0	0	0	0	0	0
Implement the logic showing SMP file in PDF format of the selected activity		Sonia, Lorenzo		2	0	0	0	0	0	2	0	0	0	0
User Story #5	2	Alessandra, Lorenzo, Sonia, Martina												
Update the DB adding for each activity the workspace notes		Lorenzo, Martina		0,5	0	0,5	0	0	0	0	0	0	0	0
Add a test area to the web page		Alessandra, Lorenzo		0,5	0	0	0	0	0	0,5	0	0	0	0
Implement test area's logic		Alessandra, Lorenzo, Sonia, Martina		3	0	0	0	0	0	1	2	0	0	0
Implement the logic showing the current workspace notes of the selected activity		Sonia, Martina		1	0	0	0	0	0	0	1	0	0	0
ESTIMATED TOTAL				40,25	5	7,75	4	3	4	7,5	3	2	3	1

Per il completamento del primo Sprint era stata effettuata una stima originaria di 37 ore. Le ore effettivamente spese per portare a termine il primo Sprint, ovvero l'*effective effort* si è discostato di poco più di 3 ore. Il Burndown Chart è un ulteriore artefatto che è un buon indicatore per apprendere la velocità del team. E' frequentemente aggiornato e mostra pubblicamente il lavoro rimanente nell'attuale Sprint Backlog.



Si evince che a livello di monte ore il team ha speso leggermente in più. Fino al sesto giorno di lavoro l'andamento è stato circa omogeneo rispettando le stime. In seguito, l'*effective effort* si è abbassato andando ad intersecare l'*estimated effort*.

Ogni giorno, come previsto dalla metodologia SCRUM, è stato effettuato il Daily Scrum, in cui il Team ha programmato la giornata di sviluppo. Nonostante la lontananza, il team ha utilizzato strumenti di messaggistica istantanea e la piattaforma Google Meet per "incontrarsi". Questo ha incentivato la collaborazione e la sincronizzazione del lavoro, fedelmente al principio dell'insieme coordinato di persone auto-gestito e cross-funzionale.

Mediante l'utilizzo della bacheca Trello e checklist, ogni membro del team per tutta la durata della progettazione ha avuto piena consapevolezza di quanto svolto e del lavoro restante.

Al termine del primo Sprint, lo SCRUM team è stato in grado di produrre un incremento di prodotto funzionante caratterizzato dalla realizzazione dei primi due mockup dell'SRS. Ogni release ha previsto una Sprint Review, in cui il team ha presentato l'incremento realizzato. Questa riunione è stata fondamentale, oltre a dimostrare quanto svolto, per iniziare a raccogliere gli input per le successive riunioni di Sprint Planning.

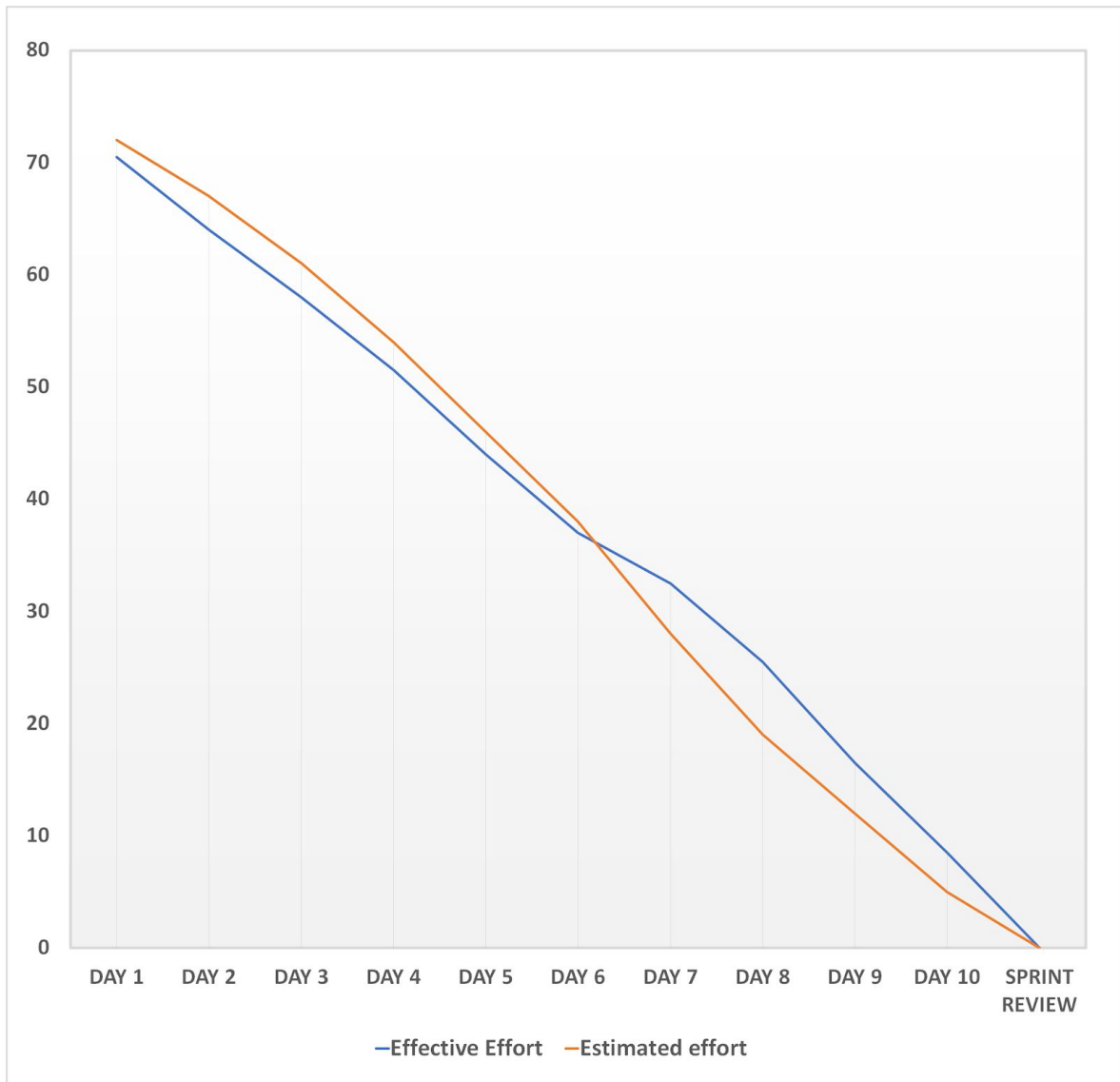
Dopo la prima Sprint Review, il team si è focalizzato sulla Sprint Retrospective: è stato un momento di dialogo e di analisi in cui sono stati individuati dei miglioramenti da essere attuati per il secondo Sprint. Il team ha preso coscienza di effettuare una stima migliore o quanto più precisa delle ore da dedicare ad ogni task.

Di seguito è riportato un estratto dello Sprint Backlog relativo al secondo Sprint.

BACKLOG TASK & ID	STORY POINTS	ASSIGNED TO	STATUS	ORIGINAL ESTIMATE	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	DAY 8	DAY 9	DAY 10
User Story#9	8	Alessandra, Lorenzo, Sonia, Martina												
Update the DB for the maintenance and their skills		Alessandra, Martina		0.5	0.5	0	0	0	0	0	0	0	0	0
Create the web page (GUI)		Sonia, Lorenzo		1	1	0	0	0	0	0	0	0	0	0
Implement web page's logic backend		Lorenzo, Sonia		5.5	2.5	0	3	0	0	0	0	0	0	0
Implement web page's logic frontend		Alessandra, Martina		5	2	2	1	0	0	0	0	0	0	0
Testing phase		Martina, Sonia, Alessandra, Lorenzo		3	0	0	0	3	0	0	0	0	0	0
User Story#8	5	Alessandra, Lorenzo, Sonia, Martina												
Create the web page (GUI)		Sonia, Alessandra		1.5	0	0	0	0	0	0	0	1	0.5	0
Implement the logic to view the skills of each maintenance		Lorenzo, Martina		2	0	0	0	0	0	0	0	1	1	0
Testing phase		Martina, Sonia, Alessandra, Lorenzo		2	0	0	0	0	0	0	0	1	1	0
User Story#10	7	Alessandra, Lorenzo, Sonia, Martina												
Update the code in DB for the maintenance adding their availability		Alessandra, Lorenzo		1.5	0.5	0	1	0	0	0	0	0	0	0
Implement the logic to select the day in which the maintenance is available		Sonia, Martina		2	0	2	0	0	0	0	0	0	0	0
Testing phase		Martina, Sonia, Alessandra, Lorenzo		3	0	0	0	0	0	0	0	0	0	3
User Story#12	5	Alessandra, Lorenzo, Sonia, Martina												
Create the web page (GUI)		Alessandra, Martina		2	0	0	0	0	0	0	0	2	0	0
Implement web page's logic		Sonia, Lorenzo		2	0	0	0	0	0	0	0	0	2	0
User Story#11	3	Alessandra, Lorenzo, Sonia, Martina												
Update the DB for the maintenance's working hours		Sonia, Alessandra		1	0	0	0	1	0	0	0	0	0	0
Create the web page (GUI)		Lorenzo, Martina		3	0	0	0	2	1	0	0	0	0	0
Implement the logic to select the list of available cars		Alessandra, Lorenzo, Sonia, Martina		5	0	0	0	0	2	1	2	0	0	0
Testing phase		Martina, Sonia		2	0	0	0	0	0	0	2	0	0	0
User Story#1	5	Alessandra, Lorenzo, Sonia, Martina												
Update the DB in order to manage cars		Alessandra, Sonia		0.5	0	0	0.5	0	0	0	0	0	0	0
Create the web page (GUI)		Martina, Sonia		2	0	0	0	0.5	1.5	0	0	0	0	0
Implement web page's logic		Martina, Lorenzo		3	0	0	0	0	1	0	0	0	0	0
Testing phase		Alessandra, Lorenzo		2	0	0	0	0	0	0	0	2	0	0
User Story#12	8	Alessandra, Lorenzo, Sonia, Martina												
Implement the logic to send a notification to the selected maintenance profile with a copy by email to the Production manager		Alessandra, Lorenzo, Sonia, Martina		12	0	2	1	1	1.5	1.5	2	1	1	1
User Story#13	8	Alessandra, Lorenzo, Sonia, Martina												
Update the DB in order to manage EWO		Sonia, Martina		2	0	0	0	0	0	0	1	0	1	0
Create the web page (GUI)		Alessandra, Lorenzo		2	0	0	0	0	0	0	0	1	0.5	0.5
Implement the logic in order to modify activity's information specific to the SWO		Alessandra, Lorenzo, Sonia, Martina		3	0	0	0	0	0	0	0	0	1	2
Testing phase		Sonia, Martina		2	0	0	0	0	0	0	0	0	0	0
ESTIMATED TOTAL				70.5	4.5	6	6.5	7.5	7	4.5	7	9	8	8.5
ACTUAL PARTIAL				70.5	64	58	51.5	44	37	32.5	25.5	16.5	8.5	

Relativamente al secondo Sprint, è stata effettuata una stima originaria di 72 ore. Le ore effettivamente spese per portare a termine il secondo Sprint sono poco più 70. Questo evidenzia che sono stati applicati i miglioramenti proposti nell'evento Sprint Retrospective. In questo caso, il Burndown Chart mostra un andamento pressoché

parallelo alla curva stimata. Fino a metà Sprint, il team ha dimostrato essere leggermente più rapido rispetto alle aspettative per poi incrementare l'*effort* allo scopo di migliorare l'architettura software, le principali viste e applicare i pattern.

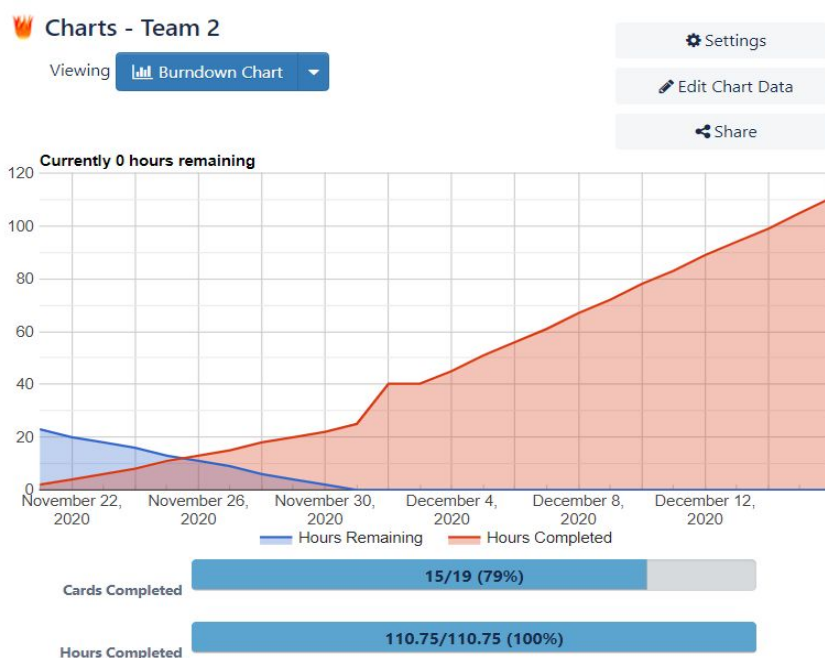
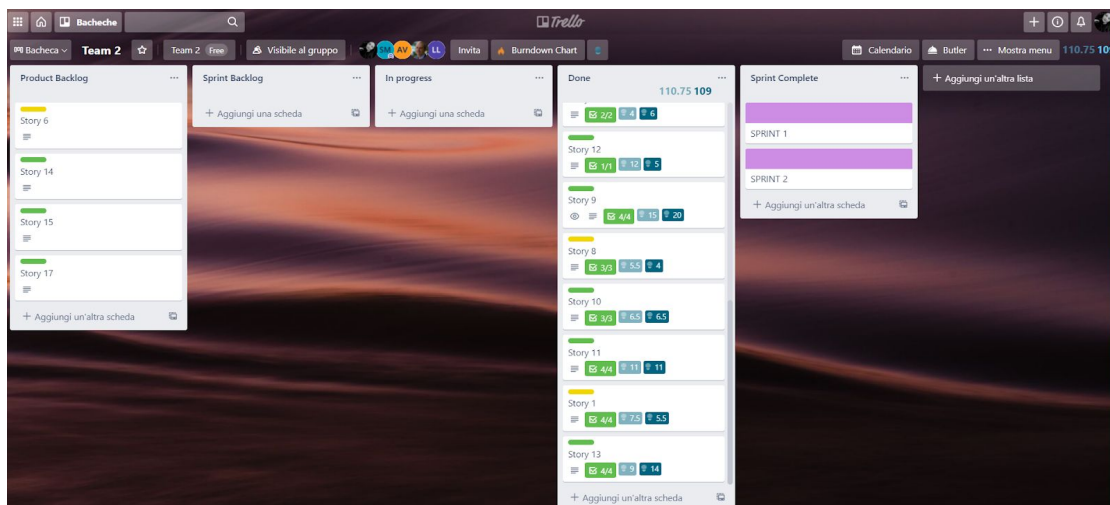


Come si evince dalla figura sovrastante, il numero di ore utilizzate per la realizzazione è stato pressoché lo stesso delle ore pronosticate. Anche il lavoro svolto ha avuto un *trend* totalmente in linea con le attese; in particolare le due curve si intersecano durante il sesto giorno di lavoro, il che lascia presagire che fino a quel giorno il lavoro è stato in linea con le attese, dopo quel giorno si era preventivato di lavorare qualche ora in più, ma evidentemente il lavoro procedeva già spedito.

1.4 Trello

Parallelamente allo sviluppo degli artefatti è stata aggiornata la bacheca Trello.

Trello è un tool per il Project Management che struttura il lavoro in bacheche condivise. Il *workspace* di Trello si divide in bacheche, che a loro volta contengono le liste, le quali sono formate da schede: questa struttura ad albero consente di avere una organizzazione molto semplice del lavoro, ma al contempo permette di tenere traccia di ogni singolo elemento. Questo tool ha permesso ad ogni componente del team di restare sempre aggiornato e abbattere la barriera della “lontananza”: in tal modo il team è sempre stato fedele al pilastro di SCRUM della trasparenza.



Capitolo 2

Progettazione

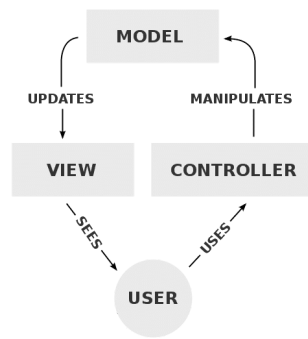
Nel seguente capitolo viene mostrata l'architettura software su cui si è basata la realizzazione di *Smart Maintenance App*, le tecnologie scelte per l'implementazione e la struttura del database.

2.1 Architettura Software

Ai fini della realizzazione di una buona progettazione e della garanzia di una struttura solida, si è inizialmente pensato a suddividere la struttura in due *sides*, in due *end-point*: uno di *front-end*, ossia la parte visibile all'utente di un programma e con cui egli può interagire (solitamente una *User Interface*), e uno di *back-end*, ossia il *side* che permette l'effettivo funzionamento di queste interazioni. Il *front end*, consente l'acquisizione dei dati di ingresso e la loro elaborazione con modalità conformi a specifiche predefinite e invarianti, tali da renderli utilizzabili dal *back end*. Il collegamento del *front end* al *back end* è un caso particolare di interfaccia.

Per l'architettura del software è stato scelto il pattern architetturale Model View Controller. Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- il *model* fornisce i metodi per accedere ai dati utili all'applicazione;
- il *view* visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
- il *controller* riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.



Il pattern MVC

Questo schema implica anche la tradizionale separazione fra la logica applicativa, a carico del controller e del model, e l'interfaccia utente a carico del view.

Il disegno architetturale è stato costruito sulla base del documento relativo alla specifica dei requisiti del software (SRS). Innanzitutto, sono stati individuati gli utenti che usufruiscono dell'applicativo:

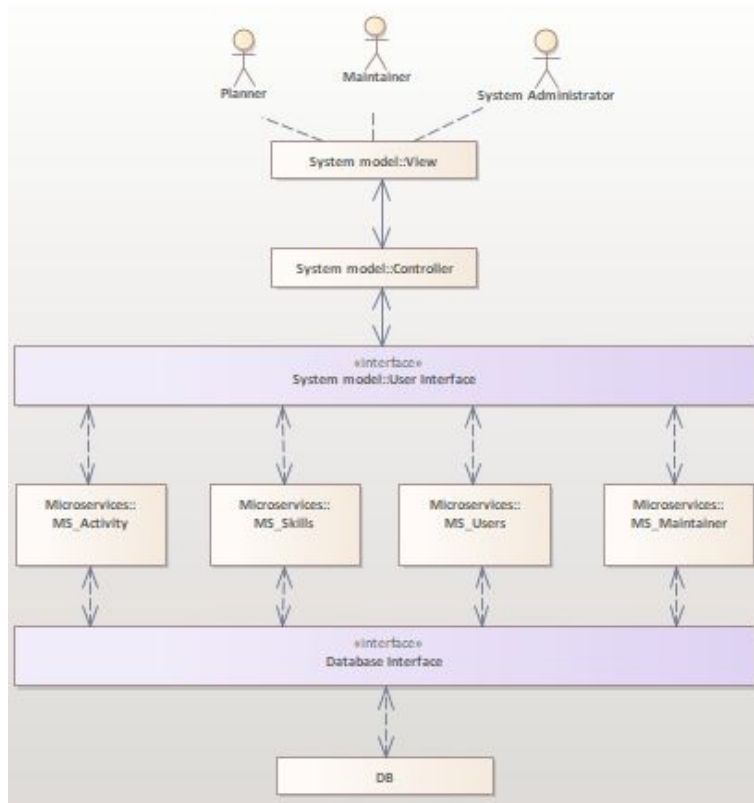
- Il *Planner*
- Il *Maintainer*
- Il *System Administrator*

Tali utenti accedono all'applicazione mediante un'interfaccia grafica, la *view*. La logica di front-end dell'applicativo è gestita dal *controller*. Infine, l'interazione con la banca dati, ossia in gergo la logica di back-end, è gestita dal *model*. Le funzionalità offerte dall'app sono state suddivise in diversi *Micro-Servizi*:

1. Micro-servizio riferito alle operazioni sulle attività presenti in banca dati;
2. Micro-servizio riferito alle operazioni sulle *skills* presenti in banca dati;
3. Micro-servizio riferito alle operazioni sugli utenti presenti in banca dati, generalmente utilizzato dalle operazioni effettuate dal System Administrator;
4. Micro-servizio riferito alle operazioni sui manutentori presenti in banca dati.

Si è scelto di implementare un unico *Database*, da cui i micro-servizi prelevano i dati, in modo tale che questi possano interagire tra loro scambiandosi dati.

Inoltre, vi è lo strato *Data Service Interface* mediante il quale si può accedere ai micro-servizi che prelevano i dati.

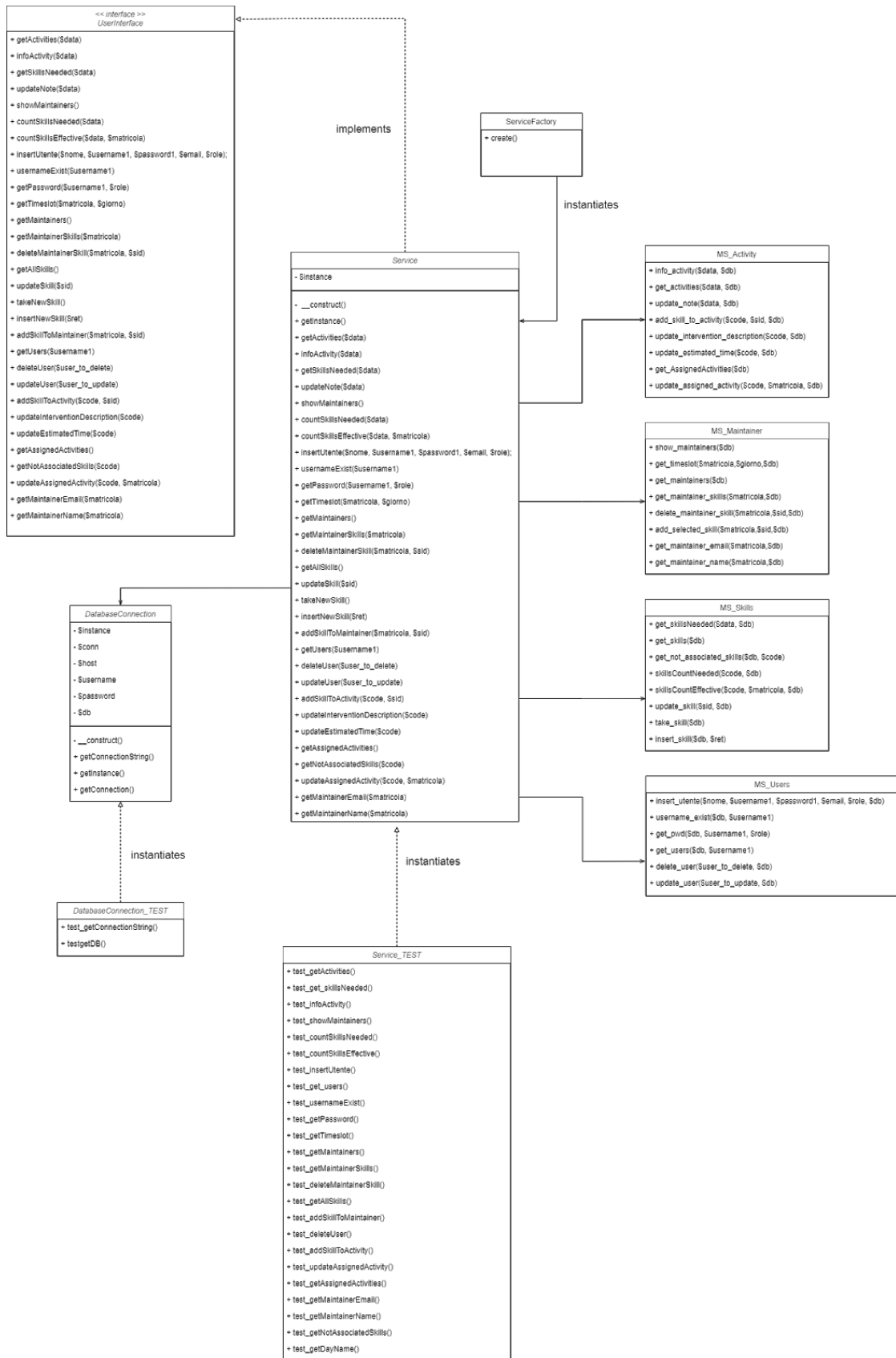


Architettura Software

Si evince dalla figura e dall'architettura precedentemente illustrata che sono stati raggiunti gli attributi di una buona progettazione software. L'obiettivo della System Decomposition è ridurre la complessità mentre si verificano cambiamenti. A tale scopo, i sottosistemi dovrebbero avere alta coesione e basso accoppiamento. In particolare, si ha coesione sequenziale in quanto l'output di una parte è usato come input della parte successiva. In più è stata raggiunta coesione funzionale siccome le parti di un modulo contribuiscono ad una singola e ben definita attività.

Per raggiungere il basso accoppiamento è stato introdotto un *layer* intermedio che separa il database dagli altri sottosistemi. Un database centralizzato condiviso avrebbe, invece, introdotto un forte accoppiamento.

2.2 Diagramma delle classi UML



2.3 Pattern applicati

Ai fini di una buona progettazione si è deciso di utilizzare dei design pattern appropriati.

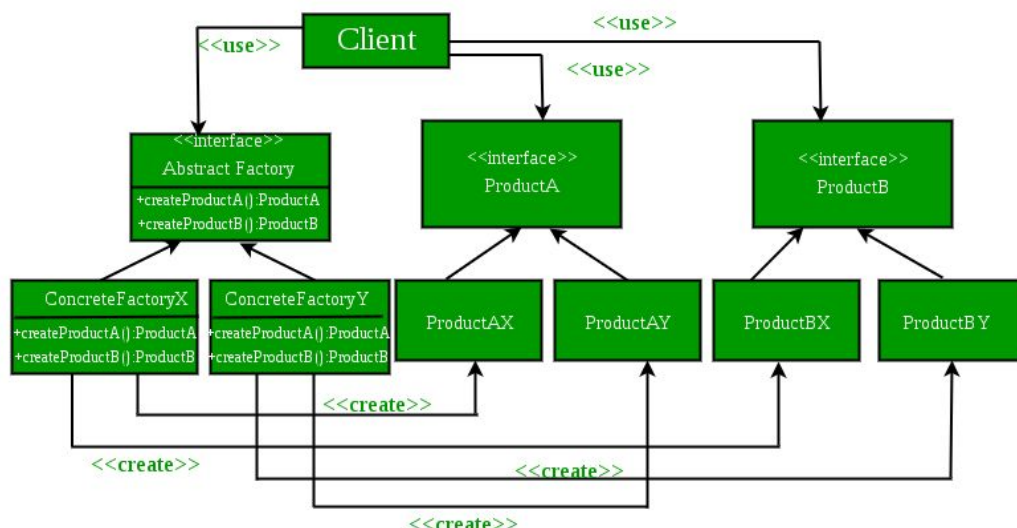
Il principale è, di certo, l'*MVC pattern*, un modello che propone una buona struttura solida e modulare all'architettura software ed è basato sulla "stretta collaborazione" tra i vari componenti, quali Model, View e Controller.

Gli altri pattern utilizzati sono:

- *Factory Pattern*, uno dei più utilizzati, rende il codice altamente manutenibile, in quanto viene implementata una classe "istanziatrice" che crea un oggetto della classe che si vuole usare;
- *Singleton*, molto utilizzato per lo sviluppo di web application, è un pattern pensato concettualmente e architetturalmente per consentire l'accesso a una e una sola istanza di una determinata classe;

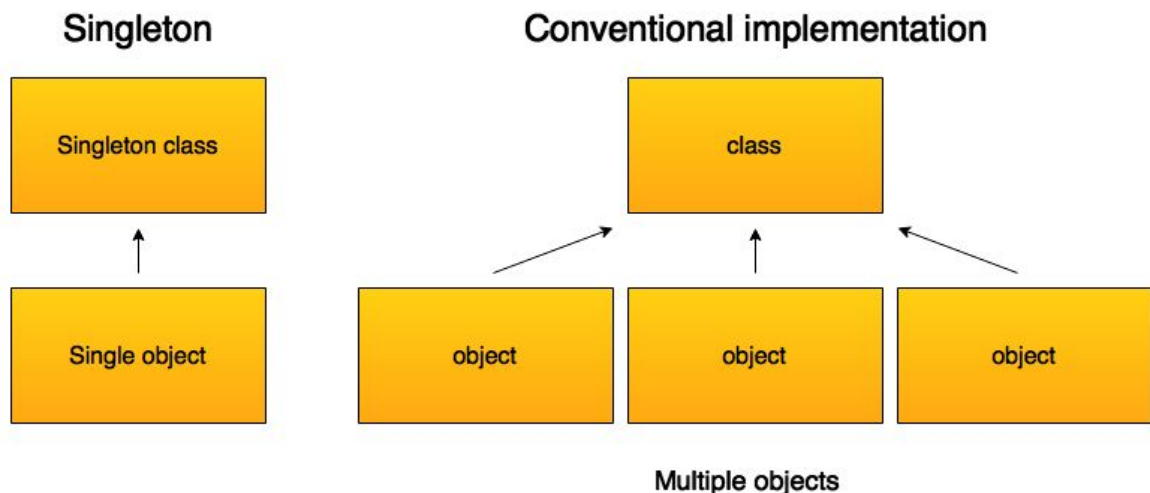
Per implementare il primo pattern, è stato necessario realizzare una classe "Factory", il cui suo unico metodo è static e restituisce un'oggetto della classe principale.

Per quanto riguarda *Factory Pattern*, i vantaggi vertono principalmente nell'alta manutenibilità del codice. Difatti, se in futuro si vuol modificare o aggiungere metodi alla *main class*, non è necessario andare a modificare ogni riga di codice in cui si è istanziati la *main class*, ma solo le righe di codice della classe "Factory". Altro benefit è, sicuramente, che nel caso in cui l'istanziamento di un oggetto della classe fosse particolarmente complesso, si può eseguire questo frammento di codice all'interno del Factory, per evitare una riscrittura dello stesso codice.



Per implementare, invece, il secondo pattern è stato necessario realizzare un metodo `getInstance`, che restituisce un'istanza della classe che lo implementa. L'utilizzo di tale pattern è risultato significativo ai fini della visibilità dei metodi; dunque, al costruttore della classe che utilizza tale pattern, è stata fornita una visibilità di tipo `private`, per evitare istanziammenti “dannosi”.

```
public static function getInstance()  
{  
    if (!self::$instance) {  
        self::$instance = new DatabaseConnection();  
    }  
  
    return self::$instance;  
}
```



2.4 Tecnologie utilizzate

Come ambiente di sviluppo è stato utilizzato XAMPP, ossia una piattaforma software che facilita l'installazione e la gestione degli strumenti di sviluppo di applicazioni web, raggruppandoli in un unico luogo. Attraverso l'installazione di XAMPP è stato installato anche il *web server* Apache, il programma che gestisce le richieste che arrivano da un qualsiasi client, nel nostro caso, attraverso PostgreSQL, cioè il DBMS, e PHP.

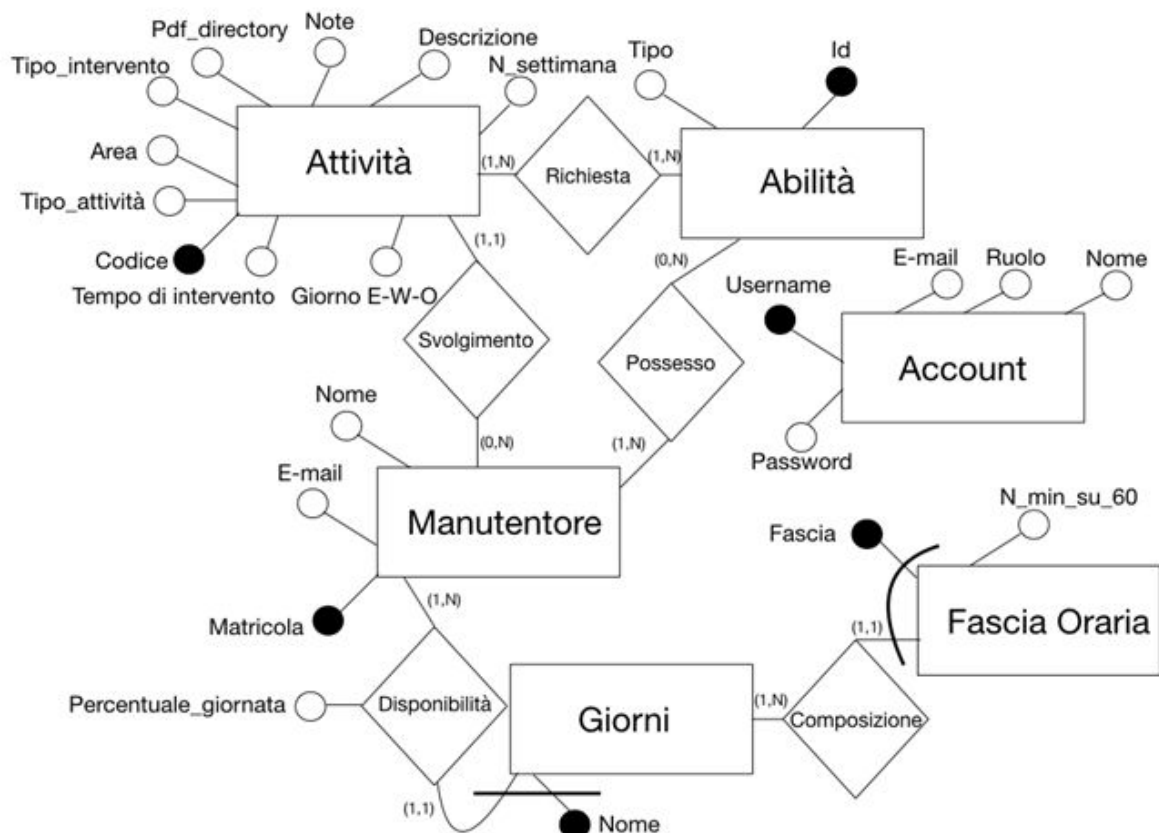
Per realizzare l'interfaccia utente sono stati utilizzati i principali linguaggi di sviluppo per applicazioni web, ossia HTML, CSS e JavaScript. Invece, come linguaggio lato server è stato utilizzato PHP, il linguaggio di scripting per la realizzazione di pagine web dinamiche.

2.5 Database

In tale sezione sono evidenziate le fondamenta di una buona web application: la creazione di un database ricco, flessibile. Le principali *utilities* del database vertono principalmente nel prelievo dei dati per poterli manipolare e mostrare all'utente; nell'inserimento, modifica e cancellazione di dati all'interno della banca. Si è deciso di utilizzare il DBMS PostgreSQL, un ottimo strumento *open-source*, già precedentemente utilizzato negli studi durante il ciclo di studi triennale. Tale scelta si è rivelata sublime, in quanto ha consentito di ottimizzare i tempi di approccio ad un nuovo ambiente, che si sarebbe potuto rivelare, inizialmente, ostile.

Le basi di una creazione di un buon database - è noto - risiedono in un'accurata progettazione concettuale che ha visto il concepimento di 6 entità, opportunamente associate tra loro, e 5 associazioni. Dunque, di seguito, lo schema Entità-Relazione:

DIAGRAMMA E-R

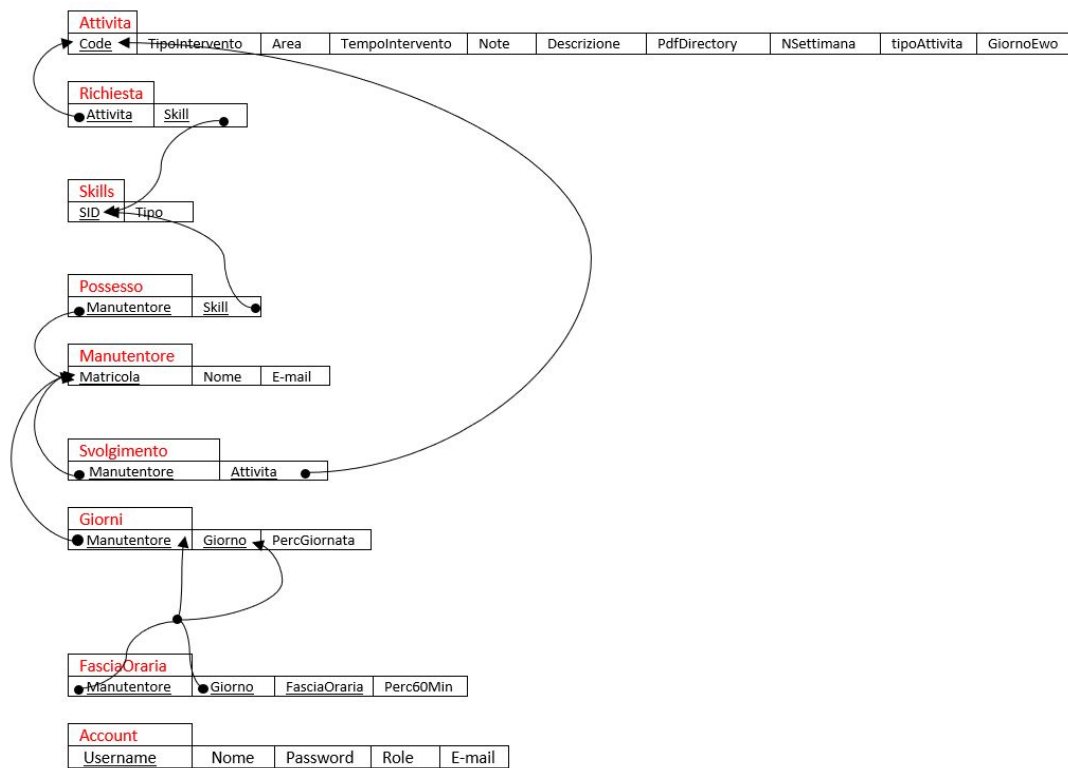


Dopo il concepimento del diagramma E-R, si è proceduto con lo stilare le principali operazioni da effettuare sul database per realizzare e implementare le principali funzionalità dell'applicazione. D'altronde sono state stilate le regole aziendali, utili ai fini del corretto utilizzo e funzionamento dell'applicazione. Per implementare alcune regole aziendali sono stati realizzati: una funzione, scritta in un linguaggio procedurale, denominato `pgpsql` e fornito direttamente da PostgreSQL, e un *trigger* che attivasse tale funzione in determinate condizioni. In particolare, al momento dell'inserimento delle attività in banca dati, il *trigger* si attiva, lanciando una procedura che controlla alcuni parametri, quali ad esempio il corretto inserimento di un'attività di tipo EWO, dunque un'attività da eseguire in condizioni di emergenza, il cui giorno di esecuzione deve essere specificato.

Successivamente, ci si è avviati verso la cosiddetta progettazione logica che può esser suddivisa in due fasi salienti: ristrutturazione dello schema E-R, con output finale uno schema totalmente ristrutturato e traduzione verso lo schema logico, con output lo schema logico definito secondo il modello scelto, quello relazionale.

Dunque, dopo aver eliminato le ridondanze, analizzato e risolto tutte le associazioni tra le relazioni, si è giunti a un primo importante risultato: lo schema logico.

SCHEMA LOGICO

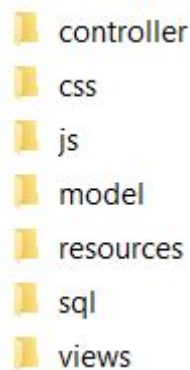


Capitolo 3

Sviluppo

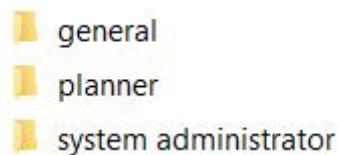
3.1 Struttura dell'app

I file sorgenti prodotti sono suddivisi in diverse cartelle, a seconda dell'utilità e del formato. La cartella *src* si presenta con le seguenti sottocartelle:

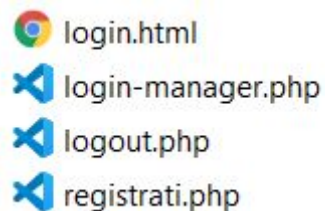


Cartelle dell'app

Nella cartella *views* sono presenti 3 cartelle che separano le pagine dedicate al Planner da quelle dedicate al System Administrator, la cartella comune *general* è dedicata alle operazioni di autenticazione.

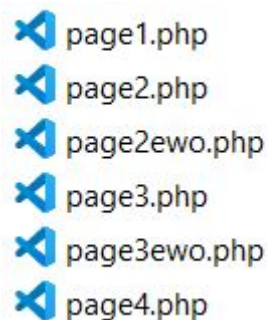


Nella cartella *general* sono presenti i seguenti file:



- `login.html`: è la schermata di benvenuto dell'app e permette l'autenticazione agli utenti che hanno già effettuato precedentemente la registrazione.
- `registrati.php`: è il file concepito per la registrazione di un nuovo utente;
- Una volta inserite le credenziali, il sistema ne verifica la correttezza e permette di raggiungere la pagina `login-manager.php`: tramite un apposito link è possibile accedere ai contenuti riservati specifici per il ruolo inserito.
- Durante la navigazione dell'app è sempre possibile, tramite un apposito bottone, effettuare il logout: nel file `logout.php` è stata concepita una schermata di arrivederci.

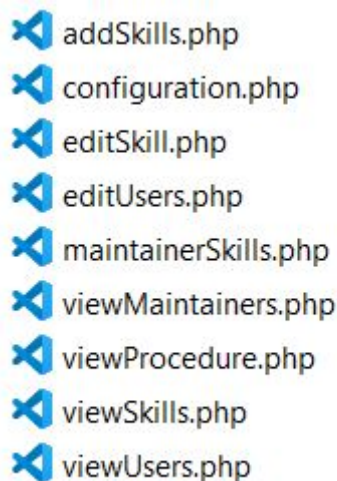
Nella cartella *planner* sono presenti i seguenti file:



- `page1.php`: pagina principale dell'interfaccia del Planner che mostra la tabella delle attività di manutenzione a seconda della settimana selezionata dall'utente;
- `page2.php`: pagina di visualizzazione dei dettagli relativi all'attività di manutenzione che deve essere svolta (descrizione dell'intervento, note, competenze necessarie, file SMP);
- `page3.php`: pagina di visualizzazione dei giorni disponibili dei manutentori;
- `page4.php`: pagina per la selezione delle fasce orarie in cui il manutentore selezionato è disponibile per svolgere l'attività;
- `page2ewo.php`: se in `page1.php` il Planner seleziona un'attività EWO (quindi non pianificata) viene trasferito in questa pagina in cui deve inserire il tempo stimato di intervento, la descrizione dell'attività e selezionare le *skills* richieste per svolgerla.

- `page3ewo.php`: pagina di selezione della fascia oraria in cui il manutentore dovrà svolgere l'attività.

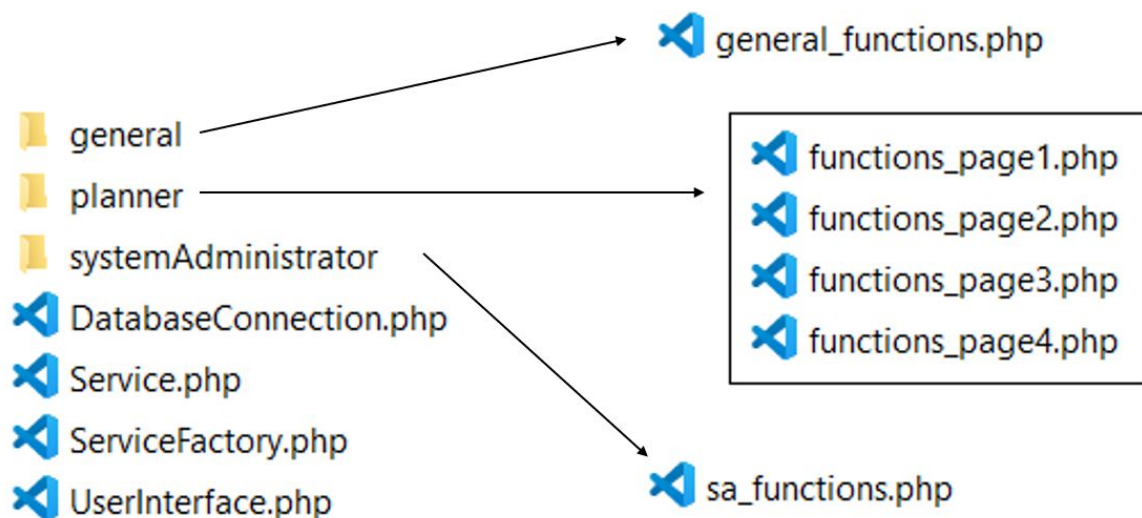
Nella cartella *system administrator* sono presenti i seguenti file:



- `configuration.php`: una volta che il System Administrator si è autenticato, questa pagina consente di visualizzare il menù principale in cui il System Administrator può selezionare un'operazione;
- `viewMaintainers.php`: pagina in cui il System Administrator visualizza i manutentori e può decidere di effettuare delle modifiche premendo il bottone "SELECT";
- `maintainerSkills.php`: pagina che consente al System Administrator di eliminare una *skill* del manutentore selezionato oppure decidere di assegnare una nuova competenza premendo il bottone "ASSIGN A NEW SKILL";
- `addSkills.php`: pagina di visualizzazione di tutte le *skills* presenti nel database. Il System Administrator, mediante le checkbox, seleziona nuove competenze al manutentore precedentemente selezionato e conferma premendo il bottone "ADD";
- `viewSkills.php`: il System Administrator gestisce le *skills* e può modificare premendo sul bottone "EDIT" e inserire una nuova competenza.
- `editSkill.php`: pagina che consente di modificare la descrizione di una competenza;

- `viewProcedure.php`: pagina che associa ad ogni manutentore l'attività che gli è stata assegnata in base alle sue competenze e disponibilità. In tal modo il System Administrator prende visione dell'organizzazione delle attività;
- `viewUsers.php`: pagina di visualizzazione di tutti gli utenti del sistema. Il System Administrator può aggiungere o eliminare un nuovo utente tramite appositi bottoni oppure decidere di modificare le proprietà di un utente premendo il bottone "SELECT";
- `editUsers.php`: pagina che consente di modificare il ruolo e l'username dell'utente.

La cartella *controller* contiene i file relativi alla connessione con il database e ai micro-servizi:

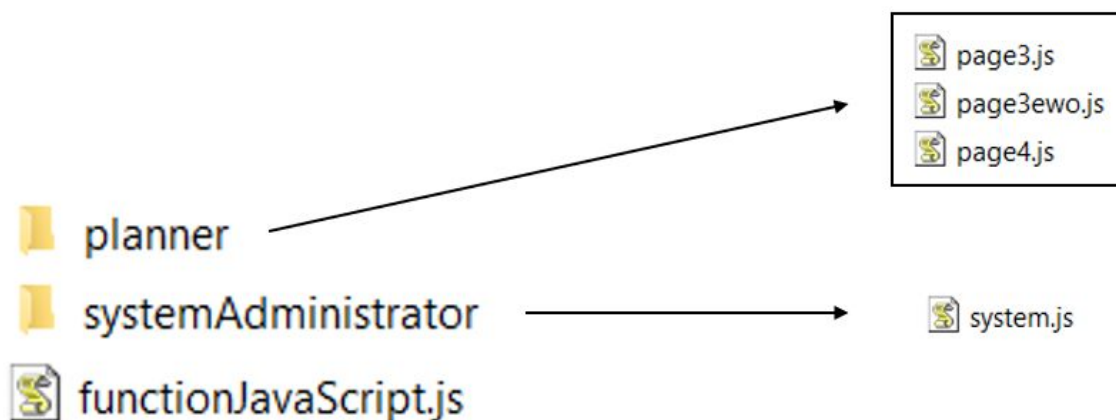


- `DataBaseConnection.php`: classe di funzioni per la connessione al database;
- Cartella *planner*, contenente i file `functions_pageX.php`: insieme di funzioni di front-end utilizzate dalla `pageX` relative all'interfaccia del planner;
- Cartella *general*, contenente il file `general_functions.php`: insieme di funzioni di *utility*;
- Cartella *systemAdministrator* contenente il file `sa_functions.php`: insieme di funzioni di *front-end* utilizzate nelle pagine relative all'interfaccia del System Administrator;

- `Service.php`: classe che implementa l'interfaccia `UserInterface` contenente tutti i servizi offerti dall'app;
- `ServiceFactory.php`: classe “istanziatrice” che crea un oggetto della classe `Service`;
- `UserInterface.php`: interfaccia che contiene tutti i prototipi dei metodi che sono implementati nella classe `Service`;

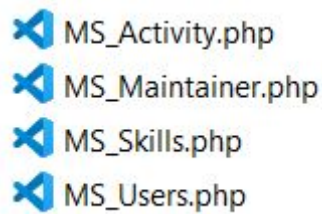
In `css` è presente il file `styles.css` che rappresenta il foglio di stile dell'applicazione.

La cartella `js` contiene i seguenti file:



- `functionJavaScript.js`: contiene le funzioni comuni a tutte le pagine, dedicate alla dinamicità di esse;
- `page3.js`: contiene le funzioni dedicate alla dinamicità di pagina tre, tra cui la logica implementata per le diverse colorazioni delle celle della tabella dipendenti dalla percentuale di disponibilità del Maintainer ;
- `page3ewo.js`: il file definisce il comportamento della pagina 3 relativa ad un'attività EWO;
- `page4.js`: il file definisce le funzioni dedicate alla dinamicità della pagina 4, tra cui la funzione dedicata all'invio di e-mail;
- `system.js`: contiene le funzioni per alla dinamicità della pagina dedicata al System Administrator;

La cartella *model* contiene i seguenti file:

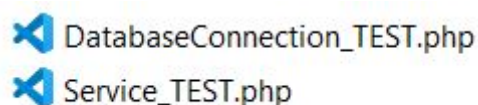


- `MS_Activity.php`: classe che raffigura il micro-servizio dedicato alle attività. Sono presenti i metodi contenenti richieste al database come ottenere le informazioni dell'attività, l'aggiornamento delle note, del tempo stimato di intervento etc;
- `MS_maintainer.php`: classe dedicata al micro-servizio per la gestione dei manutentori. I metodi effettuano query per richiedere informazioni come il nome, l'email del manutentore, la disponibilità per svolgere un'attività, cancellare o aggiungere competenze di un particolare manutentore etc;
- `MS_Skills.php`: classe per il micro-servizio riguardante le competenze. Mediante le funzioni proposte è possibile effettuare delle query al db per aggiornare le competenze, inserirne di nuove oppure richiedere il conteggio delle *skills* richieste da una particolare attività e quelle effettivamente possedute dal manutentore selezionato;
- `MS_Users.php`: classe che raffigura il micro-servizio dedicato alla gestione degli utenti. I metodi proposti consentono di interfacciarsi al db per effettuare l'inserimento/cancellazione di nuovi utenti, la verifica delle credenziali, l'aggiornamento di proprietà di un particolare utente etc;

In *resource* sono presenti le immagini e le icone illustrate dall'app e i file pdf delle *Standard Maintenance Procedure*.

Nella cartella *sql* è presente il file `db_maintenance_activity.sql` che contiene il database e il popolamento delle relative tabelle.

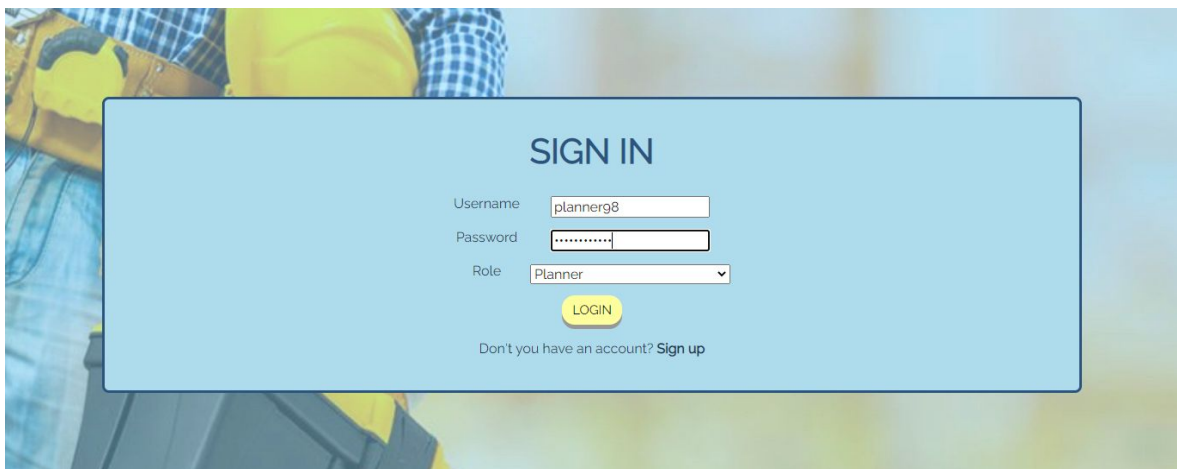
Oltre alla cartella *src* è presente la cartella *test* dedicata alle classi di test:



- `DatabaseConnection_TEST.php`: classe di test dedicata al database;
- `Service_TEST.php`: classe di test relativa alla classe `Service`.

3.2 Navigazione dell'app

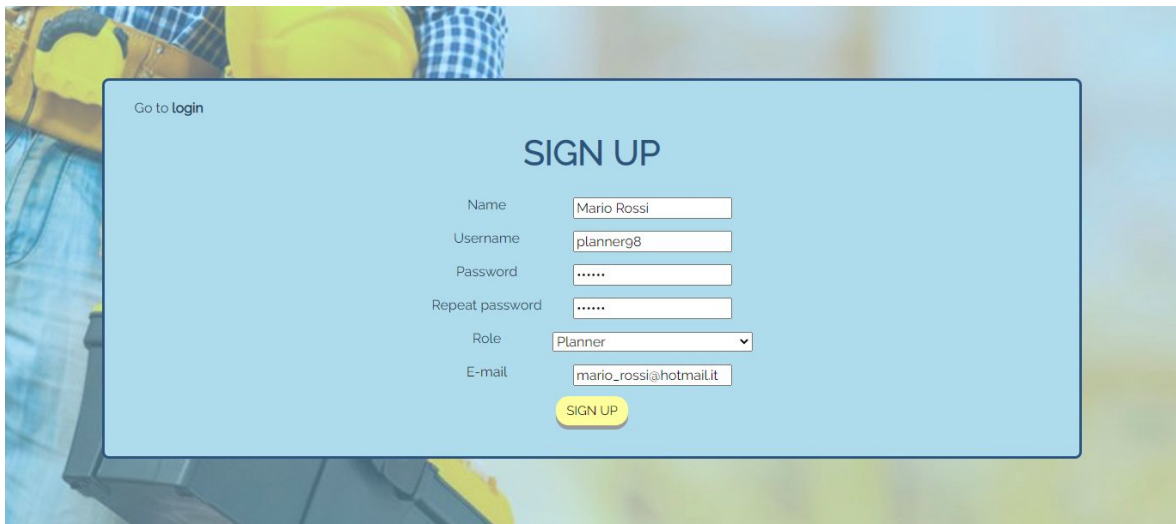
In questa sezione vengono mostrate le mappe di navigabilità della *web application*. Smart Maintenance App è caratterizzata da un'interfaccia semplice, minimale ed estremamente intuitiva. La pagina iniziale dell'applicazione è `login.html`, attraverso la quale l'utente accede alla sezione ad egli riservata inserendo *username*, *password* e *ruolo* svolto (System Administrator - Planner - Maintainer).

The image shows a login form titled "SIGN IN" overlaid on a background image of a person in a blue plaid shirt and a yellow hard hat. The form has three input fields: "Username" with the value "plannerg8", "Password" with masked characters ".....", and "Role" with a dropdown menu showing "Planner". Below the fields is a yellow "LOGIN" button. At the bottom of the form, there is a link that says "Don't you have an account? Sign up".

Se un utente non registrato prova ad accedere gli viene mostrato un messaggio di errore come quello seguente:

The user plannerg8 does not exist and/or he is not a Planner. Try again!

La creazione di un nuovo account avviene attraverso la pagina `registrati.php` a cui si accede cliccando sulla scritta *Sign up* presente nella pagina iniziale.



Go to login

SIGN UP

Name

Username

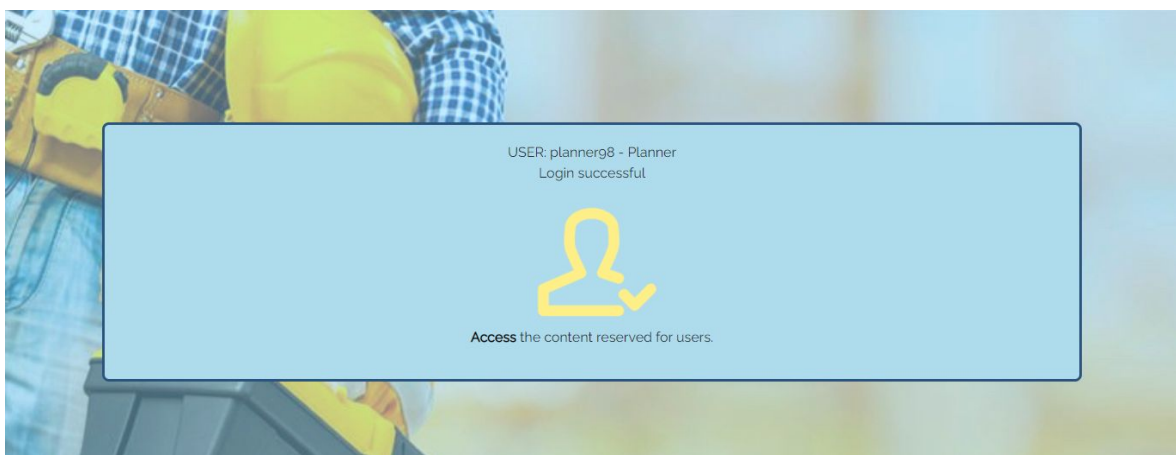
Password

Repeat password

Role

E-mail

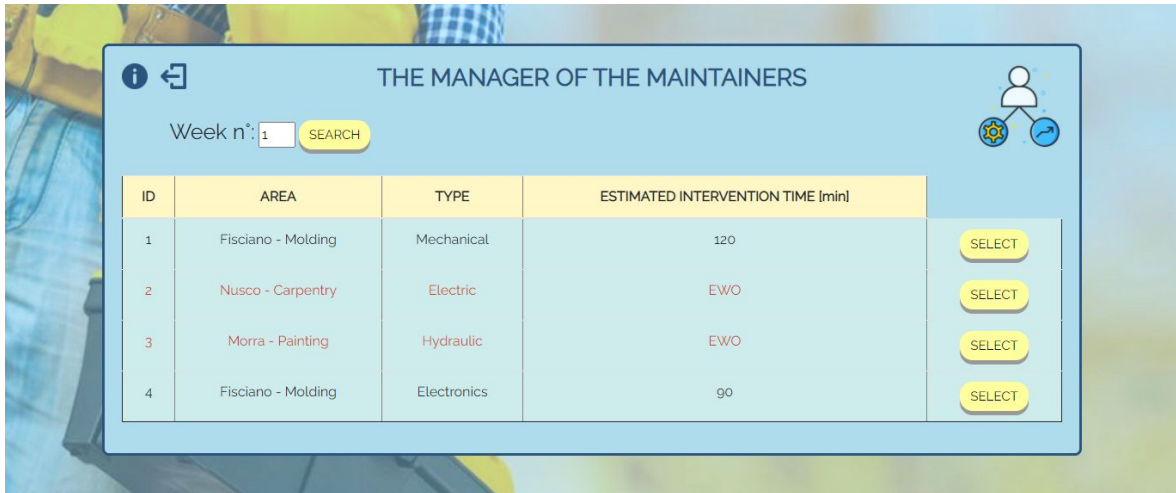
Se il login avviene con successo l'utente può accedere ai contenuti dell'applicativo cliccando su *Access*, nella pagina `login-manager.php`.



Interfaccia del Planner

La pagina principale relativa al Planner è `page1.php`. Attraverso tale pagina il Planner può visualizzare le attività di manutenzione organizzate per settimana, selezionando il numero di settimana tramite l'apposito pulsante. Una volta scelto il numero, viene mostrata la tabella delle attività contenenti per ognuna le seguenti informazioni:

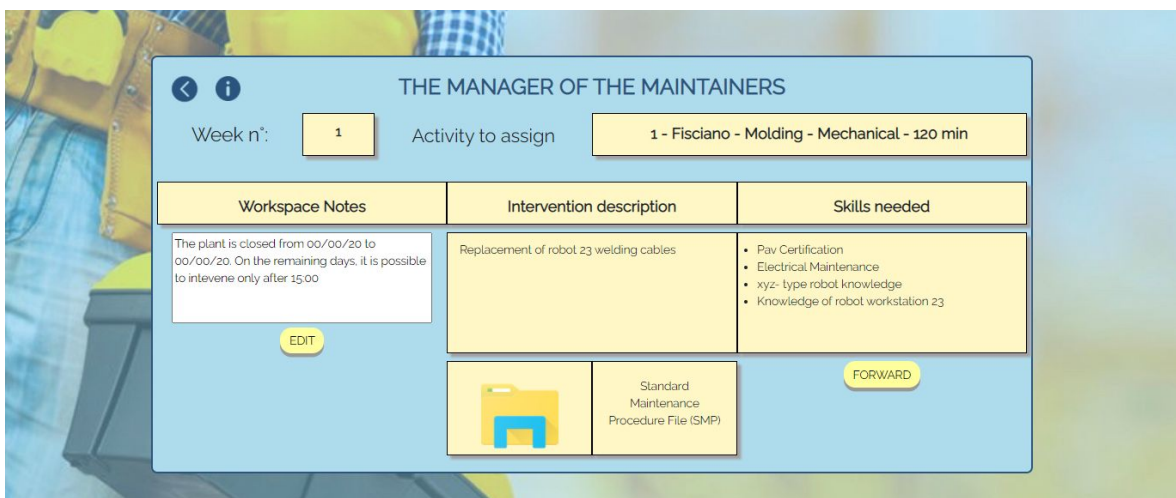
- L'ID che identifica univocamente l'attività;
- L'area di (o il dipartimento) in cui è richiesta l'attività;
- La tipologia dell'attività di manutenzione (*electrical*, *electronic*, *hydraulic*, *mechanical*);
- Una stima del tempo richiesto per l'intervento da effettuare.



Ogni riga presenta un bottone “SELECT” al click del quale si apre la pagina `page2.php` o `page2ewo.php` a seconda che l’attività sia *planned* o *unplanned* (EWO). Entrambe le pagine mostrano le informazioni salienti relative all’attività specifica, selezionata dall’utente, ossia:

- le note relative all’attività (possono essere modificate dal Planner);
- la descrizione dell’intervento che deve essere svolto;
- le skills necessarie per svolgere l’attività;
- il documento SMP (*Standard Maintenance Procedure*) in formato pdf, il quale descrive i passi da effettuare per svolgere una determinata attività di manutenzione.

Assignment of a planned activity



La pagina `page2.php` permette al Planner di verificare che le informazioni relative all'attività da svolgere siano corrette. Una volta che una specifica attività è stata verificata, il Planner può assegnare l'attività ad un Maintainer, selezionandolo in base alla sua disponibilità, espressa in percentuale, tra i Maintainer mostrati nella tabella presente in `page3.php`. A tale pagina si accede tramite il pulsante "FORWARD". Le informazioni illustrate sono: *week number*, *activity to assign* (ID, area, type, estimated intervention time), *skills needed*. La lista dei Maintainers mostra per ognuno le seguenti informazioni:

- il nome del Maintainer
- le *skills* (effettivamente possedute/richieste)
- la percentuale di disponibilità (per ogni giorno, da Lunedì a Domenica)

Il Planner può visualizzare le percentuali di disponibilità dei diversi giorni della settimana di tutti i Maintainer che possiedono almeno una competenza tra quelle necessarie per svolgere l'attività specifica. Al click di una cella l'utente è indirizzato a `page4.php`, in cui si visualizza:

- il giorno selezionato;
- il tempo richiesto per svolgere l'attività;
- le note;
- la disponibilità del Maintainer, espressa in minuti per ogni ora del suo giorno lavorativo (es. 8:00-9:00, 9:00- 10:00).

THE MANAGER OF THE MAINTAINERS

Week n°: **1** Activity to assign: **1 - Fisciano - Molding - Mechanical** Required TIME: **120 min**

Tuesday **31** Assigned TIME: **0 min** TIME to be assigned: **120 min**

Workspace Notes: The plant is closed from 00/00/20 to 00/00/20. On the remaining days, it is possible to intervene only after 15:00

AVAILABILITY Amelia Hill **40%**

Maintainer	Skills	Availab. 8:00-9:00	Availab. 9:00-10:00	Availab. 10:00-11:00	Availab. 11:00-12:00	Availab. 12:00-13:00	Availab. 14:00-15:00	Availab. 15:00-16:00	Availab. 16:00-17:00	Availab. 17:00-18:00	Availab. 18:00-19:00
Amelia Hill	3/4	0 min	0 min	0 min	0 min	0 min	0 min	60 min	60 min	60 min	60 min

SEND

Per assegnare l'attività al Maintainer il Planner deve selezionare almeno uno slot temporale e premere il bottone "SEND" affinché il Maintainer sia notificato, tramite e-mail, sull'attività assegnatagli.

A: **ameliahill@hotmail.com**

Cc: **production.managerscrum2020@gmail.com**

Oggetto: **Assignment of a planned activity**

Firma: **Nessuna**

Dear Amelia Hill,
 according to your availability, I have assigned to you the intervention with the following characteristics:
 Area - Fisciano - Molding
 Type - Mechanical
 Note - The plant is closed from 00/00/20 to 00/00/20. On the remaining days, it is possible to intervene only after 15:00
 Description of the intervention - Replacement of robot 23 welding cables
 Assigned time -120 min
 Time-slot -15:00-16:00
 16:00-17:00

Assignment of unplanned activity (EWO)

Se il Planner seleziona, invece, un'attività non pianificata, viene indirizzato alla pagina `page2ewo.php`, che mostra le informazioni relative all'attività.

THE MANAGER OF THE MAINTAINERS

Week n°: Activity to assign

Tuesday

Workspace Notes	Intervention description	Skills to add
On Monday it is possible to intervene until 6 pm, from Tuesday to Saturday until 8:30 pm	Check the car battery	<input checked="" type="checkbox"/> Pav Certification <input checked="" type="checkbox"/> Electrical Maintenance <input checked="" type="checkbox"/> xyz- type robot knowledge <input type="checkbox"/> Knowledge of robot workstation 23 <input type="checkbox"/> Experience in Machining, Fabricating, and Complex Assembly
	<input type="button" value="ADD"/>	<input type="button" value="ADD SKILLS"/>
Skills Needed	Estimated time required: <input type="text" value="45"/> min	<input type="button" value="FORWARD"/>
	<input type="button" value="EDIT"/>	

A differenza di `page2.php`, relativa ad un'attività pianificata, qui il Planner deve inserire la descrizione dell'intervento da effettuare, la stima del tempo richiesto e selezionare le *skills* necessarie per svolgere l'attività di manutenzione selezionata.

THE MANAGER OF THE MAINTAINERS

Week n°: Activity to assign

Tuesday

Workspace Notes	Intervention description	Skills to add
On Monday it is possible to intervene until 6 pm, from Tuesday to Saturday until 8:30 pm	Check the car battery	<input type="checkbox"/> Pav Certification <input type="checkbox"/> Electrical Maintenance
	<input type="button" value="ADD"/>	<input type="button" value="ADD SKILLS"/>
Skills Needed	Estimated time required: <input type="text" value="45"/> min	<input type="button" value="FORWARD"/>
<ul style="list-style-type: none"> xyz- type robot knowledge Knowledge of robot workstation 23 Experience in Machining, Fabricating, and Complex Assembly 	<input type="button" value="EDIT"/>	

Al click del pulsante “FORWARD”, viene visualizzata `page3ewo.php`, in cui è mostrata la lista dei Maintainer che possiedono almeno una *skill* necessaria a svolgere l'attività specifica e la disponibilità espressa in minuti per ogni ora del suo giorno lavorativo. Per assegnare l'attività al Maintainer è necessario raggiungere il tempo richiesto per l'intervento, selezionando gli slot temporali, in base alla disponibilità.

THE MANAGER OF THE MAINTAINERS

Week n°: 1 Activity to assign: EWO 2 - Nusco - Carpentry - Electric Required TIME: 30 min

Tuesday 31 Assigned TIME: 30 min TIME to be assigned: 0 min

Workspace Notes

On Monday it is possible to intervene until 6 pm, from Tuesday to Saturday until 8:30 pm

Maintainers AVAILABILITY

Maintainer	Skills	Availab. 8:00-9:00	Availab. 9:00-10:00	Availab. 10:00-11:00	Availab. 11:00-12:00	Availab. 13:00-14:00	Availab. 14:00-15:00	Availab. 15:00-16:00	Availab. 16:00-17:00	Availab. 17:00-18:00	Availab. 18:00-19:00
Kyle Johnson	1/2	60 min	60 min	60 min	60 min	60 min	60 min	60 min	60 min	60 min	60 min
Amelia Hill	1/2	0 min	0 min	0 min	0 min	0 min	0 min	60 min	60 min	60 min	60 min
John Butler	1/2	0 min	0 min	0 min	0 min	30 min	30 min	20 min	20 min	0 min	0 min
Wilson Fisher	1/2	5 min	11 min	60 min	60 min	11 min	11 min	11 min	11 min	60 min	60 min

Skills Needed

- Electrical Maintenance
- Experience in Machining, Fabricating, and Complex Assembly

SEND

Dopodiché, il Planner può notificare il Maintainer dell'assegnazione avvenuta inviandogli una e-mail cliccando sul tasto "SEND".

✉️ ⌵ ⏪ ⏩ 📎 📧 A 📅 📄

A: wilsonf@gmail.com ⌵

Cc: production.managerscrum2020@gmail.com ⌵

Oggetto: Assignment of an unplanned activity(EWO)

Firma: Nessuna ⌵

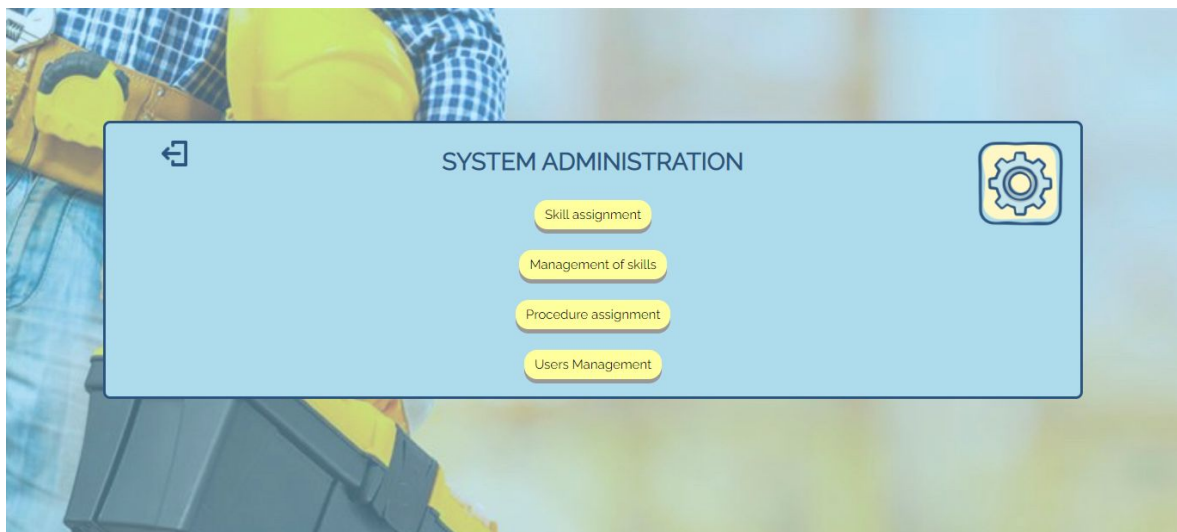
Dear Wilson Fisher,
 according to your availability, I have assigned to you the intervention with the following characteristics:
 Area - Nusco - Carpentry;
 Type - Electric;
 Note - On Monday it is possible to intervene until 6 pm, from Tuesday to Saturday until 8:30 pm;
 Description of the intervention - Change wires of the electrical system;
 Assigned Time - 230 min;
 Time-Slot:
 10:00-11:00
 11:00-12:00
 17:00-18:00
 18:00-19:00

Interfaccia del System Administrator

La pagina principale del System Administrator è `configuration.php`, che mostra all'utente i compiti che può svolgere, ossia:

- assegnare una *skill* ad un Maintainer;

- modificare o aggiungere nuove *skills* nel database;
- visualizzare i Maintainer e le attività che sono state assegnate dal Planner;
- visualizzare, modificare o eliminare gli utenti esistenti e aggiungerne di nuovi (Planner o Maintainer).



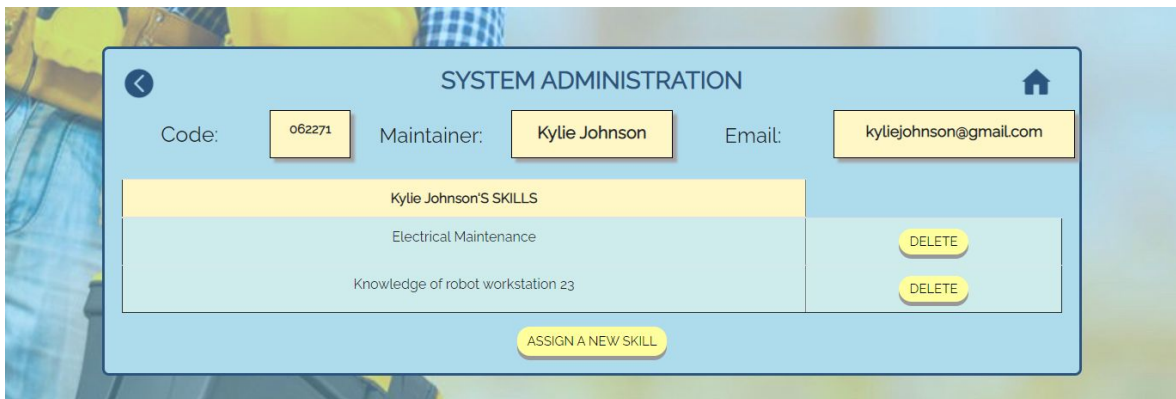
Skill assignment

Al click del pulsante “SKILL ASSIGNMENT” viene visualizzata la pagina `viewMaintainers.php` che contiene la lista di tutti i Maintainer. Per ognuno di essi è mostrata la matricola, il nome e l’e-mail associata.

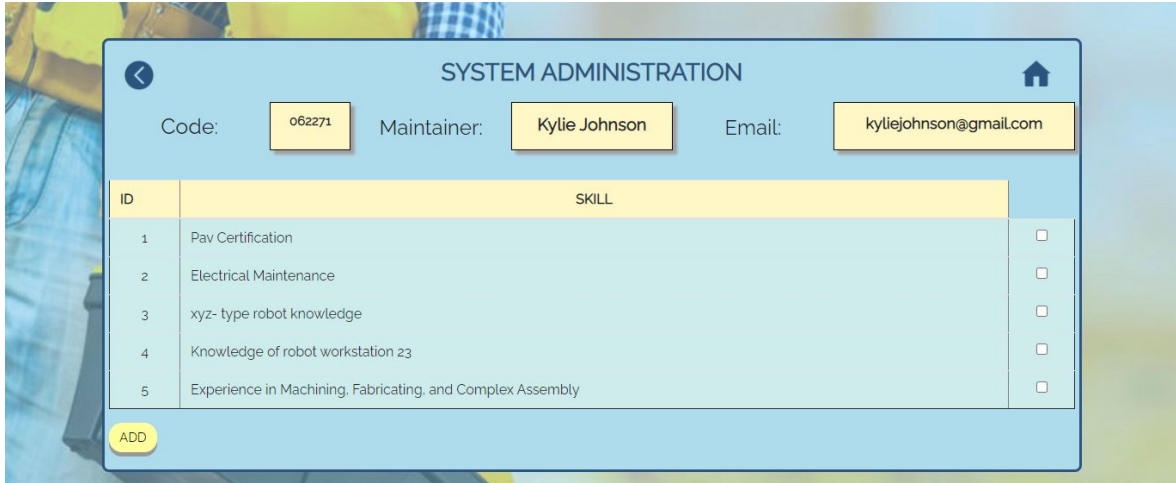
CODE	NAME	E-MAIL	
062271	Kylie Johnson	kyliejohnson@gmail.com	SELECT
062272	Amelia Hill	ameliahill@hotmail.com	SELECT
062273	Olivia Brook	oliviabrook@outlook.it	SELECT
062274	John Butler	jbutler45@gmail.com	SELECT
062275	Wilson Fisher	wilsonf@gmail.com	SELECT
062276	Harry Smith	hs45@hotmail.it	SELECT

Per visualizzare le *skills* possedute da un determinato Maintainer, l’utente deve premere il pulsante “SELECT” corrispondente alla riga della tabella associata al Maintainer. In

seguito al click, viene raggiunta la pagina `maintainerSkills.php` relativa al Maintainer selezionato. Nella seguente pagina è mostrata la tabella delle *skills* del Maintainer. L'utente può eliminare una o più *skill* premendo il bottone "DELETE". Inoltre, è possibile assegnare una nuova *skill* al Maintainer selezionato cliccando su "ASSIGN A NEW SKILL".

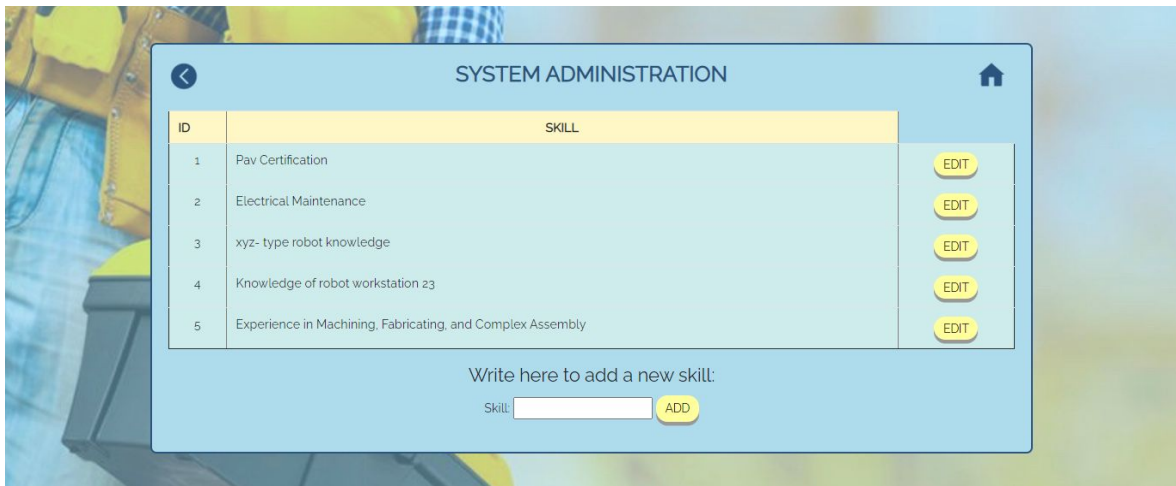


Questa azione indirizza l'utente in una nuova pagina, chiamata `addSkills.php`. Qui l'utente può selezionare una o più *skills*, da quelle esistenti, e assegnarle al Maintainer.

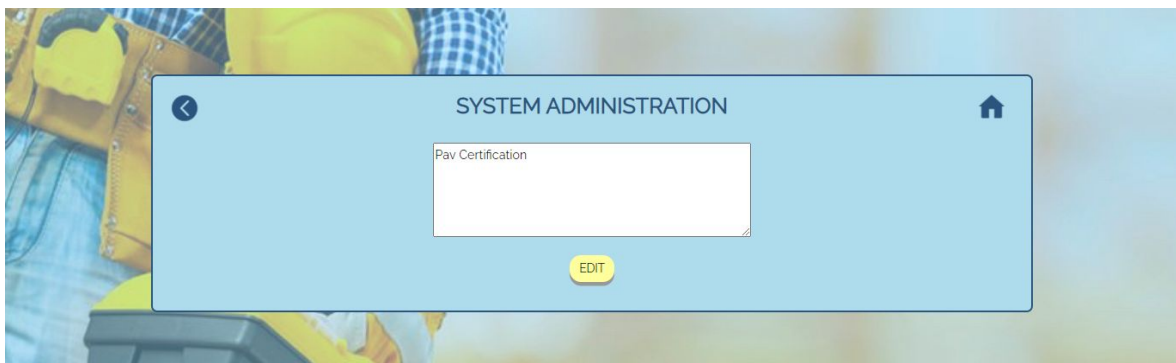


Managements of skills

Al click del pulsante "SKILL ASSIGNMENT" viene visualizzata la pagina `viewSkills.php` che mostra tutte le *skills* contenute nel database. L'utente può aggiungerne una nuova cliccando su "ADD".



Inoltre, una *skill* può essere modificata cliccando il relativo bottone “EDIT” che indirizza l’utente alla pagina `editSkill.php`, nella quale è presente una *textarea* in cui è possibile visualizzare e modificare la *skill* selezionata nella pagina precedente.



Procedure assignment

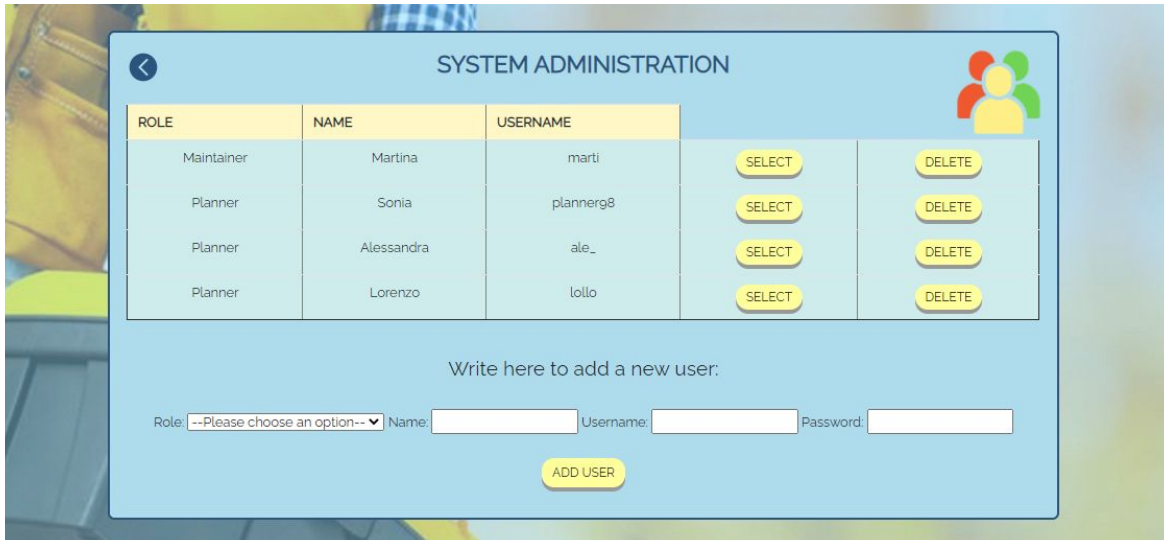
In `viewProcedure.php` l’utente può visualizzare una tabella che mostra le attività che sono state assegnate e i relativi Maintainer che la devono svolgere.

The screenshot shows a web interface titled "SYSTEM ADMINISTRATION" displaying a table of procedure assignments. The table has five columns: "MAINTAINER ID", "NAME", "ACTIVITY ID", "AREA", and "TYPE".

MAINTAINER ID	NAME	ACTIVITY ID	AREA	TYPE
062272	Amelia Hill	1	Fisciano - Molding	Mechanical
062274	John Butler	4	Fisciano - Molding	Electronics
062272	Amelia Hill	5	Salerno - Molding	Electronics
062271	Kylie Johnson	6	Fisciano - Molding	Electronics

Users management

La pagina `viewUsers.php` permette al System Administrator di visualizzare tutti gli utenti (Planner e Maintainer) esistenti. Egli può eliminare un utente cliccando su “DELETE” o aggiungere un nuovo utente riempiendo i campi del form sottostanti alla tabella e, in seguito, cliccando sul pulsante “ADD USER”.



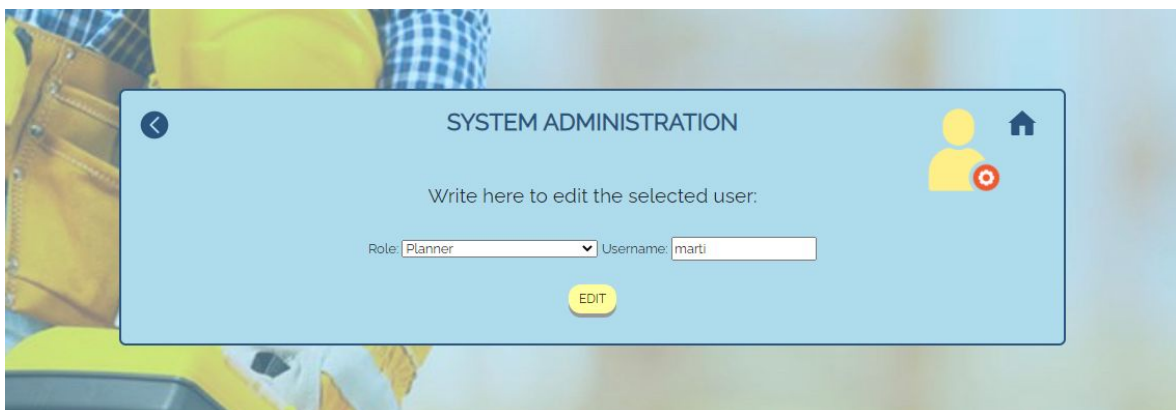
The screenshot shows a web interface titled "SYSTEM ADMINISTRATION" with a back arrow on the left and a group of three people icon on the right. It contains a table with user data and a form to add a new user.

ROLE	NAME	USERNAME		
Maintainer	Martina	marti	SELECT	DELETE
Planner	Sonia	plannerg8	SELECT	DELETE
Planner	Alessandra	ale_	SELECT	DELETE
Planner	Lorenzo	lollo	SELECT	DELETE

Write here to add a new user:

Role: Name: Username: Password:

Inoltre, è possibile modificare *ruolo* e *username* di un utente esistente cliccando sul pulsante “SELECT” relativo allo user di cui si desidera modificare uno dei due campi. Come conseguenza del clic viene visualizzata la pagina `editUsers.php` nella quale avviene la modifica effettiva dei campi.



The screenshot shows a web interface titled "SYSTEM ADMINISTRATION" with a back arrow on the left and a user icon with a red target symbol on the right. It contains a form to edit a selected user.

Write here to edit the selected user:

Role: Username:

Capitolo 4

Testing

Parallelamente allo sviluppo di codice, sono state svolte attività di testing per verificare il corretto funzionamento del codice prodotto e che l'applicazione rispettasse i requisiti. In particolare, è stata testata l'interfaccia grafica di ogni pagina e il codice di diversi metodi. È stato utilizzato il framework PHPUnit per testare il codice in linguaggio PHP.

4.1 Testing della GUI

Per testare l'interfaccia grafica è stata introdotta una tabella contenente tre colonne che riportano, rispettivamente, il requisito, la relativa descrizione del comportamento atteso, e il risultato del test effettuato. I requisiti sono stati presi dalle *user stories* dei clienti. Inoltre, le pagine dell'app sono state confrontate con i *mock-up* proposti, anch'essi dai clienti.

Testing GUI - prima release:

REQUISITO	DESCRIZIONE	RISULTATO
Visualizzazione delle attività di manutenzione ordinate per settimana	Per poter mostrare le attività di manutenzione da svolgere, è necessario prelevare i dati da un database e mostrarli tramite la pagina Web al client. Tramite apposito input, verranno visionate solo le attività della settimana selezionata dall'utente.	Superato
Visualizzazione informazioni specifica attività	In seguito a clic del tasto "Select" riferito a una specifica attività presente in Page1, l'user viene reindirizzato a Page2, dove può visionare le informazioni relative allo specifico intervento.	Superato
Visualizzazione dei diversi tipi di attività	All'interno della page1, l'user può visualizzare il tipo di ogni attività schedata.	Superato
Download del SMP file in formato PDF	All'interno della page2, l'user, mediante apposito pulsante, può scaricare l'SMP file in formato PDF relativo all'attività selezionata.	Superato
Facoltà di modifica delle Workspace Notes	All'interno della page2, l'user, può modificare le note relative all'attività selezionata, digitando la nuova nota all'interno dell'apposita area di testo, cliccando successivamente il tasto "Edit".	Superato

Testing GUI - seconda release:

REQUISITO	DESCRIZIONE	RISULTATO
Login all'applicazione	Per poter eseguire il login l'utente inserisce le proprie credenziali e in seguito al click del bottone queste vengono ricercate all'interno del database. Se sono presenti il login viene eseguito con successo.	Superato
Sign up	Un nuovo utente, che intende iscriversi alla piattaforma attraverso la pagina di sign up, compila il form inserendo i propri dati. Al click del bottone 'sign up' i dati vengono inseriti nel database	Superato
INTERFACCIA PLANNER		
Visualizzazione Maintainers	Vengono visualizzati i profili di tutti i maintainers con in allegato il loro periodo di disponibilità per giorni della settimana. I dati vengono prelevati dal db.	Superato
Selezione Maintainer	Al click di una delle celle della tabella, indicante le percentuali di disponibilità dei maintainers, il planner viene riportato al profilo del maintainer selezionato. Nella pagina in cui viene riportato vi sono gli orari di disponibilità del maintainer per fasce orarie. I dati vengono prelevati dal db	Superato
Invio E-mail al Maintainer	Il planner sceglie di assegnare al maintainer l'attività cliccando la cella con l'orario prescelto. Dopo aver selezionato l'orario il planner cliccando sul bottone send invia la mail al maintainer selezionato	Superato
INTERFACCIA SYSTEM-ADMINISTRATOR		
Skill assignment	Il System Administrator accede alla lista dei maintainers, la quale viene prelevata dal database. Selezionando uno specifico maintainer vengono mostrate le skills di quest'ultimo, anch'esse prelevate dal db. Il SA può eliminare quindi una determinata skills del maintainer, tale dato viene eliminato anche nel db	Superato

REQUISITO	DESCRIZIONE	RISULTATO
Management of skills	Il System Administrator visualizza tutte le possibili skills, tale info viene prelevata dal db. Il SA può modificare una specifica skills o aggiungere nuove skills al sistema. Alla modifica o all'aggiunta delle skills tali azioni vengono riflesse sul database	Superato
Procedure assignment	Viene mostrata la lista delle attività assegnate ad ogni manutentore. Tali dati vengono prelevati dal database	Superato
Users management	Attraverso questa interfaccia il SA può visualizzare il ruolo di ogni user presente sulla piattaforma, tali informazioni vengono prelevate dal database. Inoltre il System Administrator può modificare il ruolo degli user già presenti, aggiungere o eliminare user. Le azioni eseguite vengono riflesse sul database	Superato

4.2 Testing unitario con PHPUnit

Per il testing unitario, ossia il collaudo di singole unità software, è stato utilizzato PHPUnit un framework di *unit test* per il linguaggio di programmazione PHP.

Sono stati introdotti metodi di test per verificare il corretto funzionamento di alcuni metodi. Tali metodi sono stati suddivisi in classi di test. Affinché i test possano essere eseguiti, le classi devono estendere la classe TestCase. Tale classe deve, quindi, essere importata in ogni classe di test, tramite il seguente istruzione che utilizza la *keyword* use:

```
use PHPUnit\Framework\TestCase;
```

Testing sulla connessione al database

Per testare la connessione al database è stata introdotta la classe DatabaseConnection_TEST:

```

use PHPUnit\Framework\TestCase;
include "../src/controller/DatabaseConnection.php";

final class DatabaseConnection_TEST extends TestCase {

    public function test_getConnectionString(){
        $db = DatabaseConnection::getInstance();
        $connection_string = "host=localhost dbname=postgres user=postgres password=1234";
        //$connection_string = "host=localhost dbname=postgres user=postgres password=1235"; //-- FAIL
        $this->assertEquals($connection_string, $db->getConnectionString());
    }

    public function testgetDB() {
        $db = DatabaseConnection::getInstance();
        $this->assertNotEquals(False, $db->getDB());
    }
}

```

- **test_getConnectionString**

Il metodo serve per testare `getConnectionString`, ossia il metodo che restituisce la stringa contenente i parametri per la connessione al database. La funzione `assertEquals` verifica che la stringa restituita dal metodo da testare sia corretta.

- **testgetDB**

Il metodo verifica che la connessione al database vada a buon fine. A tale scopo, è utilizzato il metodo `assertNotEquals` che restituisce *True* se il valore di `getDB` è diverso da *False*, ossia se la connessione è avvenuta con successo.

Test Coverage sulla classe DatabaseConnection_TEST:

The screenshot displays an IDE with two tabs: `DatabaseConnection_TEST.php` and `DatabaseConnection.php`. The `DatabaseConnection.php` tab is active, showing a PHP class with private static and private attributes. The `Coverage: DatabaseConnection_TEST` panel on the right shows a table with two columns: `Element` and `Statistics, %`. The table lists several files, with `DatabaseConnection.php` highlighted, showing 93% lines covered. The `Test Results` panel at the bottom shows a summary of the test run, indicating that 2 tests passed and 2 assertions were successful.

Element	Statistics, %
DatabaseConnection.php	93% lines
functions_page1.php	
functions_page2.php	
functions_page3.php	
functions_page4.php	
general_functions.php	
sa_functions.php	
Service.php	
ServiceFactory.php	
UserInterface.php	

Tests passed: 2 of 2 tests – 80 ms

Testing started at 16:43 ...

Time: 00:00.338, Memory: 22.00 MB

OK (2 tests, 2 assertions)

Testing sui servizi

Per testare il corretto funzionamento dei micro-servizi è stata realizzata la classe `Service_TEST` che implementa diversi metodi di test.

Sono presentate due funzioni di test relative a `MS_Activity`:

- `test_getActivities()`

```
public function test_getActivities()
{
    $micro = ServiceFactory::create();
    $this->assertFalse($micro->getActivities( data: "")); //se l'utente non inserisce un input per weekNumber, la funzione restituisce false
    $activity3 = ['7', 'Hydraulic', 'Salerno - Molding', '45', 'Robot motor oil has burned out', 'It is possible to intervene only after 14:
    $ret = $micro->getActivities( data: 3);
    $row = pg_fetch_row($ret);
    for ($i = 0; $i < count($activity3); $i++) {
        $this->assertEquals($activity3[$i], $row[$i]);
    }
}
```

Questo metodo serve per testare `getActivities`, il quale restituisce le attività di manutenzione relative alla settimana selezionata dall'utente, data in input al metodo. Se l'utente non fornisce alcun input per il numero di settimana, la funzione restituisce `False`. Quindi, per verificarne il corretto funzionamento è stata utilizzata la funzione `assertFalse` sul metodo `getActivities`, che riceve come parametro una stringa

vuota: in questo caso la funzione di `assert` deve restituire *True*. In aggiunta sono state confrontate tutte le informazioni relative alle attività della settimana 3.

- **test_addSkillToActivity()**

```
public function test_addSkillToActivity(){
    $micro = ServiceFactory::create();
    $esito = $micro->addSkillToActivity( code: 1, sid: 5);
    $this->assertTrue($esito); //l'assegnazione della skill all'attività è andata a buon fine
}
```

La funzione di test verifica che si possa aggiungere una competenza ad un'attività. In particolare, la funzione in esame restituisce *True* se la query è andata a buon fine. Il metodo `assertTrue` verifica che il booleano `$esito` sia uguale a *True*.

Di seguito si descrivono alcune funzioni del micro-servizio `MS_Users`.

- **test_insertUtente()**

```
public function test_insertUtente()
{
    $micro = ServiceFactory::create();
    $ret = $micro->insertUtente( nome: 'MartinaTest', username: 'planner', password: '1234', email: 'martinalamberti3@gmail.com', role: 'Planner');
    $this->assertTrue($ret);

    $ret_empty = $micro->insertUtente( nome: '', username: '', password: '1234', email: '', role: '');
    $this->assertFalse($ret_empty);

    // -- FAIL --
    /*
    $ret_empty = $this->micro->insertUtente('', '', '1234', '', '');
    $this->assertTrue($ret_empty);
    */
}
```

La funzione di test è concepita per verificare la corretta restituzione del booleano. Il metodo da verificare restituisce false se l'inserimento nel db non va a buon fine oppure non sono stati inseriti dei campi obbligatori.

- **test_getPassword()**

```
public function test_getPassword()
{
    $micro = ServiceFactory::create();
    $ret = $micro->getPassword( username: 'planner', role: 'Planner');
    $this->assertTrue(password_verify( password: "1234", $ret));

    // - FAIL : la pass è 1234, non 1235
    // $this->assertTrue(password_verify("1235", $ret));
}
```

Quest'interessante funzione di test verifica se la password inserita nel database corrisponde a quella inserita dall'utente. Per questioni di privacy, la password è criptata nel db: a tale scopo è utilizzata la funzione `password_verify` che restituisce `true` se i due parametri forniti sono uguali.

Ora la descrizione si concentra su due funzioni di test prelevate a campione da `MS_Maintainer`.

- `test_deleteMaintainerSkill()`

```
public function test_deleteMaintainerSkill()
{
    $micro = ServiceFactory::create();
    $esito = $micro->deleteMaintainerSkill( matricola: "062273", sid: 1);
    $this->assertTrue($esito); //la cancellazione è andata a buon fine
}
```

La funzione `delete` restituisce un booleano rappresentante l'esito dell'operazione. Se la query nel db va a buon fine, la funzione restituisce `True`. Di conseguenza è stato utilizzato il metodo `assertTrue`.

- `test_addSkillToMaintainer()`

```
public function test_addSkillToMaintainer()
{
    $micro = ServiceFactory::create();
    $esito = $micro->addSkillToMaintainer( matricola: "062273", sid: 5);
    $this->assertTrue($esito); //è andato a buon fine siccome il manutentore non possiede la skill con codice 5
    $esito_neg = $micro->addSkillToMaintainer( matricola: "062273", sid: 5);
    $this->assertFalse($esito_neg);
}
```

La funzione da testare restituisce un booleano rappresentante l'esito dell'operazione. La funzione restituisce `False` nel caso in cui il manutentore selezionato abbia già la *skill* suddetta. Mediante l'utilizzo di `assertTrue` e `assertFalse`, è possibile verificare il corretto funzionamento del metodo.

Di seguito, sono riportate due funzioni di test relativamente a `MS_Skills`.

- `test_getNotAssociatedSkills()`

```

public function test_getNotAssociatedSkills(){
    $micro = ServiceFactory::create();
    $ret = $micro->getNotAssociatedSkills( code: 4);
    $row = pg_fetch_row($ret);
    $result = [3, "xyz- type robot knowledge"];
    for ($i = 0; $i < count($result); $i++) {
        $this->assertEquals($result[$i], $row[$i]);
    }
    $row = pg_fetch_row($ret);
    $result = [5, "Experience in Machining, Fabricating, and Complex Assembly"];
    for ($i = 0; $i < count($result); $i++) {
        $this->assertEquals($result[$i], $row[$i]);
    }
}

```

Il test ha lo scopo di verificare se il risultato della query restituito dalla funzione `getNotAssociatedSkills` è corretto. Preso come parametro il codice dell'attività, si vuole conoscere quali sono le competenze non richieste. Tramite `assertEquals` si confronta il valore atteso definito nella variabile `$result` con il risultato della funzione in esame.

- `test_get_skillsNeeded()`

```

public function test_get_skillsNeeded()
{
    $micro = ServiceFactory::create();
    $ret = $micro->getSkillsNeeded( data: 5);
    //devo testare se in $ret ci sono le due skills: 1) Electrical Maintenance, 2) xyz- type robot knowledge
    $row = pg_fetch_row($ret);
    $this->assertEquals('Electrical Maintenance', $row[0]);
    $row = pg_fetch_row($ret);
    $this->assertEquals('xyz- type robot knowledge', $row[0]);
}

```

Questo è il metodo implementato per testare la corretta restituzione delle competenze necessarie per svolgere un determinato intervento di manutenzione. Il test è stato effettuato sulle *skills* relative all'attività con ID = 5. In base ai dati effettivamente contenuti nel database, risulta che per tale attività le '*skills needed*' sono due:

- *Electrical Maintenance*
- *xyz – type robot knowledge*

Per verificare, quindi, che `showSkills` restituisca effettivamente queste due competenze è stata invocata la funzione `assertEquals`. Tale funzione controlla che il risultato della query, restituita dal micro-servizio, contenga due righe il cui contenuto sia equivalente alle stringhe riportate sopra.

Test Coverage sulla classe Service_TEST:

The screenshot displays the IDE interface with the `Service_TEST.php` file open. The coverage table on the right shows the following data:

Element	Statistics, %
DatabaseConnection.php	46% lines
functions_page1.php	
functions_page2.php	
functions_page3.php	
functions_page4.php	
general_functions.php	6% lines
sa_functions.php	
Service.php	76% lines
ServiceFactory.php	100% lines
UserInterface.php	

The bottom status bar indicates: **Tests passed: 24 of 24 tests - 209 ms**. Below this, it shows the command used to run the tests: `C:\xampp\php\php.exe -dxdebug.coverage_enable=1 C:\bin\phpunit.phar --coverage-clover C:\Users\martina.elle\A`. The final status is **OK (24 tests, 98 assertions)**.

Coverage: Service_TEST x

100% files, 50% lines in 'model'

Element	Statistics, %
MS_Activity.php	38% lines
MS_Maintainer.php	73% lines
MS_Skills.php	39% lines
MS_Users.php	56% lines