Corso di laurea in informatica

Progetto ingegneria del software

Documentazione esercizio 2 (passaporto)

Alberto Ondei (VR471795)
Abdullah javed (VR471543)
Riccardo Corsini (VR472109)
2022/23

Indice

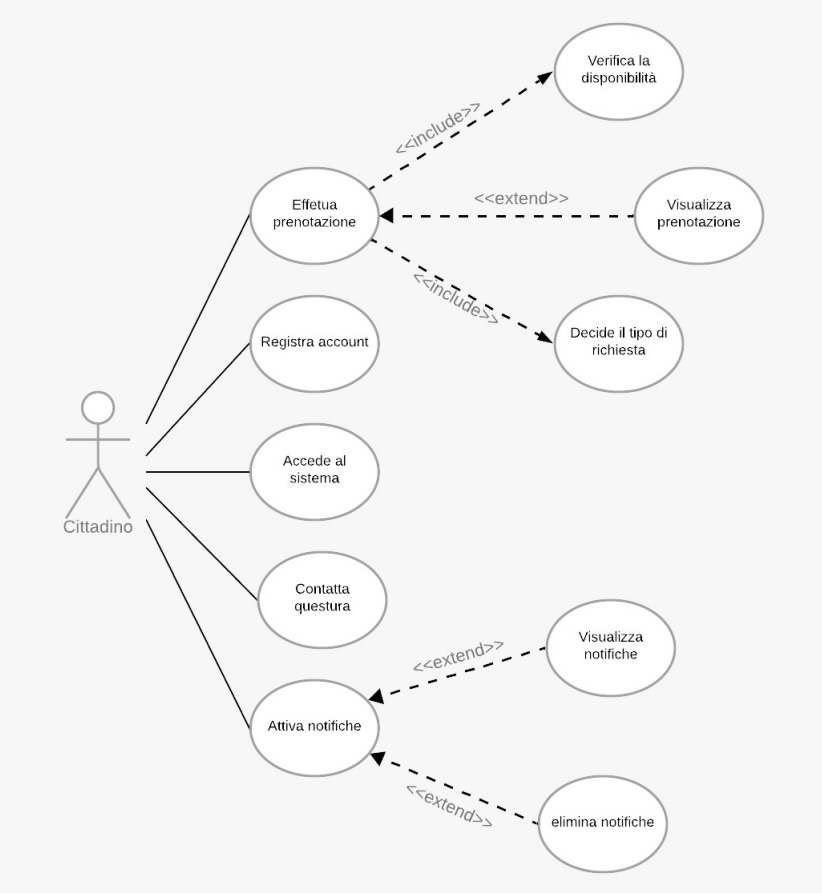
1 Requisiti ed interazione utente-sistema

1.1 Diagrammi dei casi d'uso

Note Il sistema proposto può essere usato da cittadini o dal personale della questura.

Entrambi per poter accedere ai servizi devono potersi loggare, con la differenza che i cittadini devono prima registrarsi, mentre il personale ha a disposizione delle credenziali pre-fornite dagli amministratori del sistema salvate nel database.

1.2 Diagramma casi d'uso cittadini

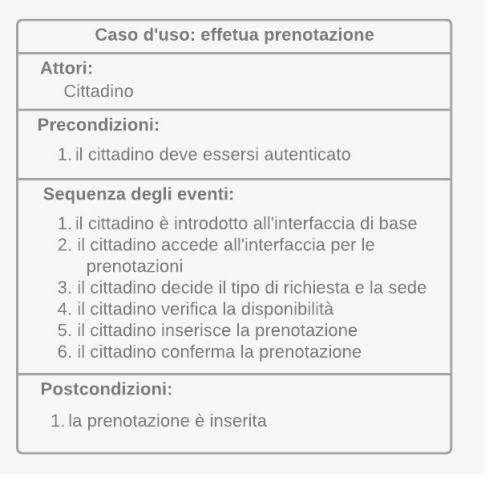


1.2.1 Specifiche casi d'uso cittadini

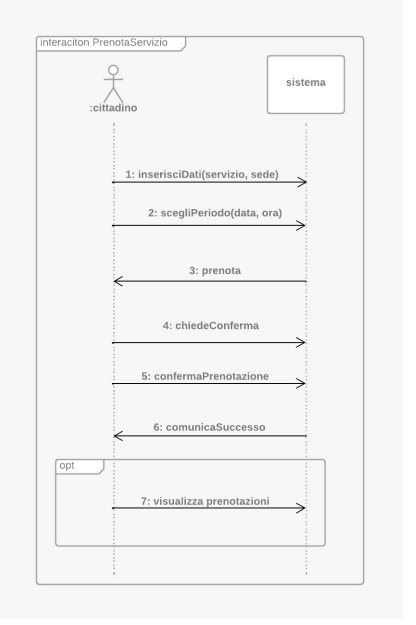
Effettua prenotazione I cittadini hanno a disposizione i servizi per i passaporti a cui possono aderire:

- prenotazione del rilascio del passaporto, con diverse modalità(per scadenza del precedente, per furto, per smarrimento, per deterioramento, e così via)
- prenotazione del ritiro del passaporto, il quale sarà disponibile solo dopo un mese dalla data di richiesta di rilascio
- nel caso in cui il cittadino abbia già prenotato un rilascio e anche un ritiro, non potrà scegliere nessun servizio fino a quando la data del ritiro non sarà passata

Per ogni servizio il cittadino può vedere gli orari e le sedi disponibili giorno per giorno e selezionare il momento desiderato presso la sede opportuna.



La linea di vita di come si evolve la prenotazione di un servizio da parte del cittadino:

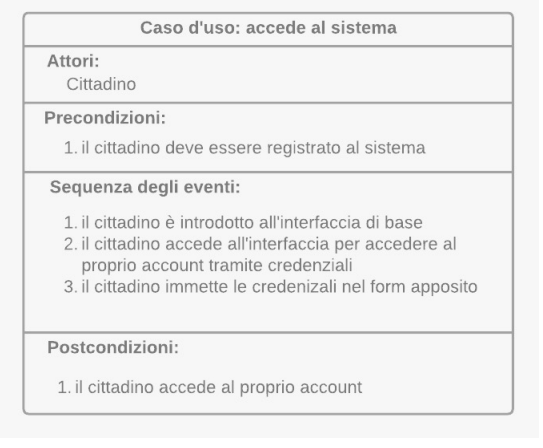


- inserisciDati(servizio,sede): inserisce il servizio che vuole prenotare e la sede specifica in cui si trova
- scegliPeriodo(data,ora): verrà fornita al cittadino una tabella in cui ci sarà un apposità legenda dei colori, le righe della tabella rappresentano le ore mentre le colonne le date e i campi della tabella saranno colorati in modo da far capire se il servizio si può prenotare, se non si può prenotare perchè i posti sono pieni o se non si può prenotare perchè non è stata messa disponibilità da parte della questura
- chiedeConferma: dopo aver inserito i dati il sistema manda un alert in cui chiede conferma
- confermaPrenotazione: diamo come presupposto che il cittadino decide di confermare, se non lo fa semplicemente rimane nella stessa schermata senza aver inserito i dati
- comunicaSuccesso: dopo aver prenotato viene mandato in una schermata che conferma la corretta esecuzione dell'operazione

- visualizza prenotazione: infine può visualizzare la prenotazione fatta nell'appositá area menu che mostra le prenotazioni effettuate

Registra account I cittadini che necessitano di un servizio legato al rilascio del passaporto possono accedere al sistema di prenotazione dopo essersi registrati.

Accesso all'account I cittadini possono accedere ai servizi riguardanti il passaporto solo dopo aver fatto l'accesso al sistema tramite le proprie credenziali decise in fase di registrazione.



Attiva notifiche Per quanto riguarda l'attivazione delle notifiche il cittadino, nella stessa schermata in cui prenota la data e l'ora, può selezionare una data e un orario non ancora aggiunti dal personale, e se lo fa quando verrà aggiunto un servizio in quell'orario il cittadino verrà notificato dal sistema nella schermata del menu apposita dove ci sono le notifiche.

Caso d'uso: attiva notifiche

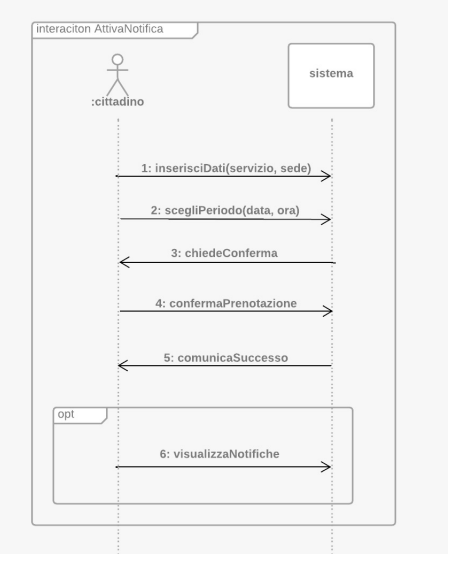
Attori:
Cittadino

Precondizioni:
1. il cittadino deve aver scelto il tipo di servizio e la sede

Sequenza degli eventi:
1. il cittadino seleziona la data e l'ora che non sono ancora state gestite dalla questura
2. il sistema chiede al cittadino se vuole attivare le notifiche per il giorno e l'ora selezionati
3. il cittadino conferma l'attivazione della notifica

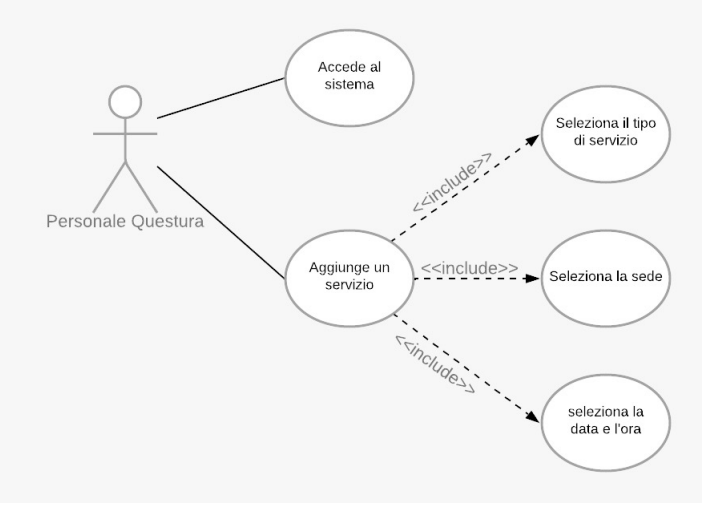
Postcondizioni:
1. la notifica per il giorno e l'ora selezionati viene attivata

La linea di vita di come si evolve l'attività di attivare la notifica da parte del cittadino è molto simile a quella per prenotare il servizio, l'unica differenza è data dal fatto che il cittadino in questo caso seleziona una data che non è stata ancora inserita dal personale in modo da riceverne la notifica.



1.3 Diagramma casi d'uso personale

NB il personale deve solo accedere al sistema perchè le credenziali gli verranno già fornite quindi non ha bisogno di registrarsi

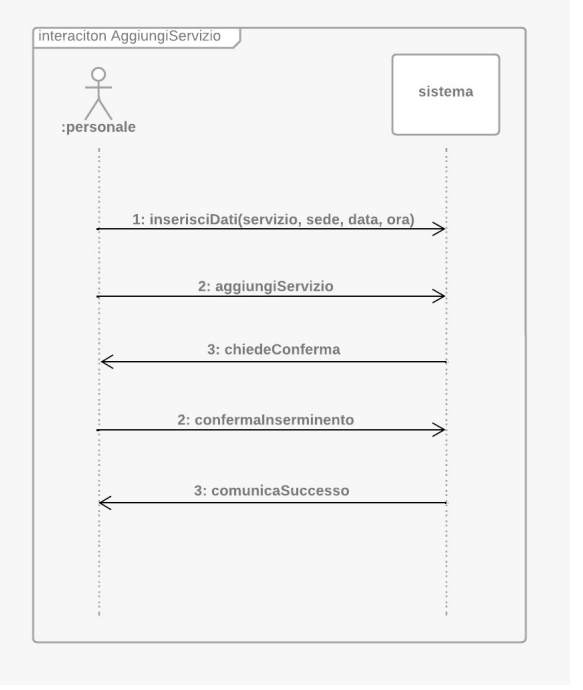


1.3.1 Specifiche casi d'uso personale

Aggiunge un servizio

Caso d'uso: aggiunge un servizio
Attori: Personale questura
Precondizioni: 1. il personale deve essersi autenticato
Sequenza degli eventi: 1. il personale è introdotto all'interfaccia di base 2. il personale seleziona il tipo di servizio 3. Il personale seleziona la sede 4. il personale seleziona la data e l'ora 5. il personale aggiunge il servizio 6. il personale conferma l'aggiunta del servizio
Postcondizioni: 1. l'inserimento è avvenuto con successo

La linea di vita di come si evolve l'aggiunta di un servizio da parte del personale:



- inserisciDati(servizio,sede,data,ora): il personale deve compilare i dati del servizio che vuole aggiungere
- aggiungiServizio: dopo aver compilato i dati schiaccia il bottone per inserire il servizio
- chiedeConferma: Il sistema quindi chiede conferma
- confermaInserimento: anche qua supponiamo che gli vada bene e che inserisce i dati confermando
- comunicaSuccesso: il sistema comunica al personale che è avvenuto tutto con successo

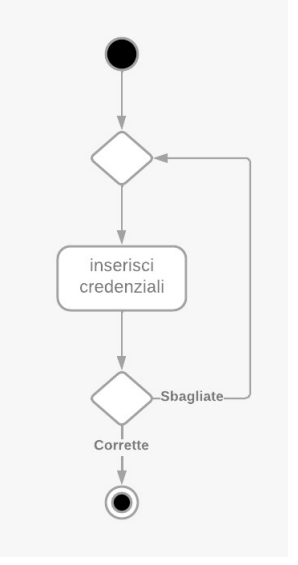
Accesso all'account L'accesso all'account per quanto riguarda il personale avviene nella stessa maniera del cittadino.

1.4 Diagrammi di attività

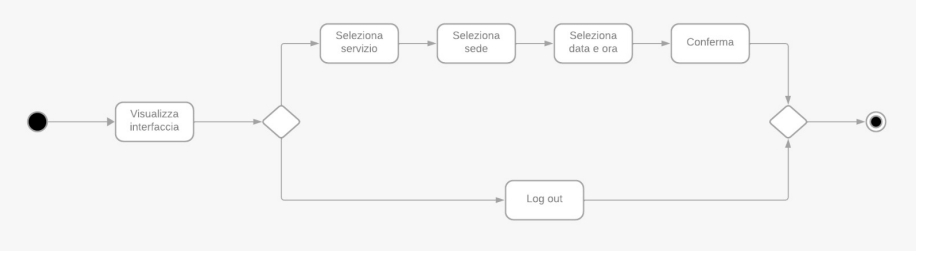
I diagrammi di attività usati servono per mostrare meglio come avviene il flusso di esecuzione delle attività sia per quanto riguarda i cittadini che per quanto riguarda il personale, tramite il passaggio dei token.

1.4.1 Diagrammi di attività personale

Il diagramma di attività per accedere al sistema tramite credenziali:



Il diagramma di attività dopo che il personale ha fatto l'accesso al proprio account e accede ai servizi:



Quindi, il personale vede l'interfaccia e può scegliere se inserire un servizio oppure fare il logout

2 Architettura e implementazione

Abbiamo adottato un metodo di sviluppo dei processi agile e incrementale perchè la pianificazione del lavoro da fare l'abbiamo decisa man mano che venivano eseguite le attività, ed inoltre la fase di specifica, sviluppo e validazione dei processi sono state fatte in modo intercalato, e abbiamo suddiviso il lavoro in piccoli incrementi in cui abbiamo lavorato su parti specifiche del software, questo ci ha permesso di modificare parti di programma che ritenevamo inadeguate e di aggiungere migliorie del programma più facilmente, e anche di risolvere problemi di incrompensione dei requisiti in modo poco drastico.

2.1 Design architetturale

Il nostro software non usa del tutto pattern architetturali visti a lezione, infatti, abbiamo usato una variante del pattern MVC in cui il controller assume un valore predominante in quanto ci sembrava più facile dividere la gestione del codice tra le varie schermate andando a creare super classi nel caso in cui parti di codice corrispondevano completamente. La nostra progettazione si avvicina quindi, al pattern architetturale MVC per quanto riguarda il controller e il view.

Infatti si può distinguere la parte del controller che in base alle interazioni fatte dall'utente cambia i dati che sono immagazzinati nel database, la view manda al controller le eventuali attività fatte dell'utente come quando seleziona determinati campi, mentre è il **controller** che interroga il database e chiede quello che deve chiedere e manda al view quello che l'utente deve vedere dopo l'interazione avvenuta col database.

Il comportamento del nostro software quindi è simile a quello di un model view controller, ma non lo rispecchia completamente in quanto non abbiamo fatto una classe model che manda le notifiche alla parte visiva(view), ma abbiamo usato un semplice database, questo ci ha permesso di avere una minore complessità di codice eliminando le classi del model.

Anche senza la classe model, il nostro controller riesce comunque a interagire tranquillamente col database e quando c'è da mandare dati dal database a view ci pensa il controller a fare da intermiediario.

2.1.1 Gerarchia delle classi

Per quanto riguarda la gerarchia delle classi abbiamo notato:

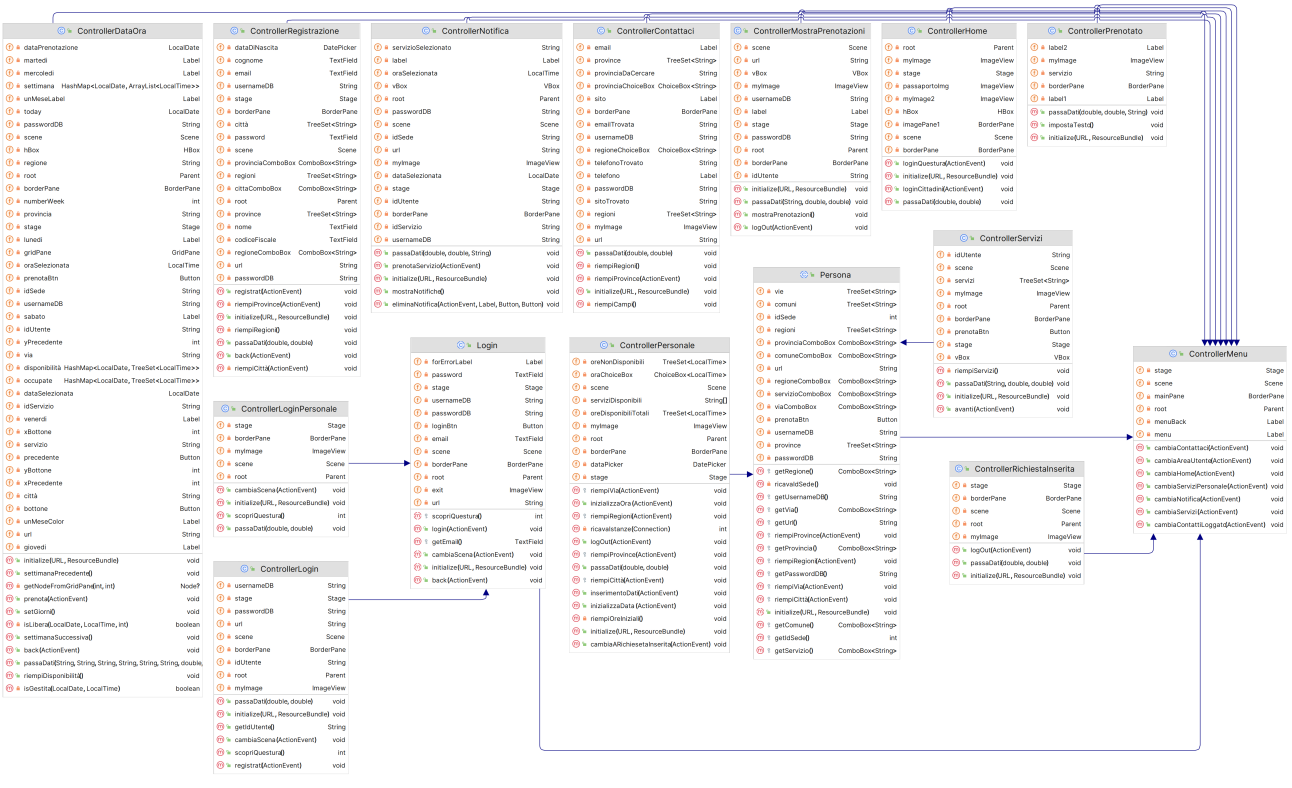
- Alivello implementativo il codice per la gestire il personale della questura e quello per gestire l'utente aveva delle uguaglianze nella parte di selezione del posto in cui dover prenotare(da parte del cittadino)/inserire il servizio(da parte del personale).

  Perchiò abbiamo fatto una superclasse persona che implementava questa parte e abbiamo evitato che ci fosse della rindondanza di codice superflua.

- Anche per quanto riguarda la parte del login utente/personale abbiamo notato che ci sarebbe stato del codice rindondate perchè dovevano fare praticamente la stessa cosa, l'interazione col databse era quasi la stessa e cambiava solo la verifica che doveva controllare se era un utente della questura o un cittadino che si stava loggando.

- Inoltre tutte le classi controller hanno in comune una superclasse menu, in quanto in tutte le scene c'è un menu che aiuta la navigazione tra scene e quindi abbiamo deciso di mettere i metodi per il cambio di scena questa superclasse che venivano ereditati dai controller.

Ecco graficamente come viene mostrato la gerarchia delle classi:



2.2 Design pattern

Oltre ai pattern iterator e altri che sono alla base della programmazione in java, abbiamo usato l'observer, questo ci è servito in più contesti:

1. L'abbiamo usato nella sezione contattaci, in quanto volevamo fare in modo che selezionata la regione e la provincia da contattare venissero mostrati la email, il numero di telefono e il sito della questura di quella provincia.

   Per farlo abbiamo usato un pattern observer in cui:

   - Come soggetto abbiamo la choicebox in cui viene inserita la provincia;
   - L'osservatore invece è l'implementazione anonima della classe ChangeListener<String>, in cui viene implementato il metodo 'changed', e ogni volta che viene cambiato il valore al soggetto (choicebox), quest'ultimo avvisa l'osservatore attivando il metodo changed che a sua volta chiama un metodo per riempire i campi e impostare le label email, telefono e sito in base alla provincia inserita.

2. L'abbiamo usato per rendere l'interfaccia responsive mettendo:

   - Le properità 'widthProperty()' e 'heightProperty()' dell'oggetto 'borderPane' come soggetti;
   - L'osservatore invece è l'implementazione di 'ChangeListener' passata come argomento al metodo 'addListener()', e questo codice viene attivato quando la larghezza o lunghezza della 'borderPane' cambia.

     Infatti quando i due soggetti (larghezza o lunghezza) vengono cambiati, il metodo 'changed' dell'osservatore viene chiamato e oggetti della scena come immagini vengono ridimensionati in proporzione alla dimensione del 'borderPane' .

   Non abbiamo usat

3 Test

Durante la fase di testing, abbiamo adottato diversi approcci al fine di garantire la qualità del nostro progetto.

Il modo in cui abbiamo testato il software si può dividere in 3 fasi:

- Il test delle componenti in cui abbiamo testato le componenti pezzo per pezzo man mano che le sviluppavamo
- Il test sul software completo in cui abbiamo testato il progetto con tutte le sue componenti riunite

- Il test fatto da terzi in cui abbiamo coinvolto:
  - Colleghi esterni al progetto per ricevere feedback da persone estranee al processo di sviluppo, al fine di ottenere suggerimenti su come migliorare sia il codice che l'esperienza utente.
  - Genitori inesperti in modo che il progetto cercasse di essere il più semplice possibile

Questo approccio ci ha permesso di individuare prontamente eventuali anomalie, anche in risposta alle specifiche del progetto.

Tra i vari test eseguiti per valutare la funzionalità e la robustezza del programma, possiamo identificare i seguenti:

1. Durante la fase di registrazione, sono stati testati i controlli per segnalare le cause degli errori di inserimento, che potevano derivare da **dati non presenti nell'anagrafica** o da **utenti già registrati nel sistema.**

2. Nella fase di login, con utenti presenti nella base di dati, inserendo correttamente l'email ma sbagliando la password e viceversa.

3. Test nella fase di prenotazione di un servizio, con particolare attenzione alle specifiche del progetto che richiedevano di permettere agli utenti di ritirare un passaporto solo dopo un mese dalla richiesta, e inoltre che abbiamo disabilitato date e orari precedenti al tempo odierno.

   Inoltre è emerso un problema, ovvero abbiamo dovuto decidere come gestire se succede che rimane una sola prenotazione disponibile per una data e due o più cittadini accedono alla schermata di prenotazione proprio di quella datà nello stesso momento.

   Abbiamo deciso che il più veloce a compilare la prenotazione riuscisse a inserire i dati mentre al più lento arriva un alert di errore.

   Inoltre in questo caso abbiamo coinvolto genitori nel test che hanno suggerito miglioramenti nell'interfaccia grafica, poiché ritenevano il processo di prenotazione poco intuitivo. Abbiamo quindi apportato modifiche, ispirandoci il più possibile alla realtà.

4. In fase di attivazione delle notifiche da parte dei cittadini, assicurandoci che il sistema gestisse correttamente le notifiche per le date previste quando un cittadino le attivava.

5. Durante la visualizzazione delle prenotazioni effettuate in precedenza dagli utenti, cliccando su mostra prenotazione ottenevamo dei dati duplicati oppure non li ottenevavamo proprio. Questo è stato risolto utilizzando cambiando un errore presente nella scrittura della query

6. Durante l'accesso da parte della questura, e' stata testata la capacità del sistema di riconoscere se stesse cercando di accedere un utente della questura o un utente normale. E' stato anche verificato che, una volta che un utente della questura inseriva un servizio, il sistema lo rendesse disponibile accedendo al sistema con un utente cittadino. Inoltre, e' stato testato il corretto funzionamento in caso di inserimento di dati errati come la comparsa del messaggio di errore "credenziali errate, riprovare!".

7. Cliccando sull'interfaccia ai collegamenti tra le diverse schermate del software, tra cui il caso in cui la schermata "Contattaci" non mostrava i contatti quando si selezionava una sede in una città. Questo problema è stato risolto allineando le basi di dati tra i membri del team.

8. Durante la prenotazione e stato testato il corretto funzionamento quando si cliccava sulla cella grigia, per gestire date non disponibili (perchè non ancora inserite dal personale), chiedendo agli utenti se desideravano ricevere una notifica quando il servizio sarebbe stato reso disponibile dalla questura.

9. Facendo testare il logout, ci siamo assicurati che dopo aver premuto il tasto "logout", l'utente fosse correttamente disconnesso e ritornasse alla schermata principale.

10. Durante la selezione dei servizi, delle regioni, delle province, dei comuni e delle sedi funzionasse correttamente e che mostrasse i comuni della provincia scelta in precedenza e che a sua volta fosse corretta rispetto alla regione scelta.

11. E' stato testto il corretto funzionamento durante la prenotazione di un servizio, abbiamo implementato una schermata pop-up di conferma, seguita da una schermata che indicava i documenti necessari da portare.

12. E' stato testato anche in modo barbarico il passaggio consecutivo tra diverse schermate per assicurarci che non ci fossero cambiamenti grafici che potessero compromettere l'esperienza utente.

13. Durante il test nella prenotazione del servizio e' sorto un problema durante la selezione della data-ora del servizio, che generava errori nelle celle selezionate

14. Ci siamo assicurati che ci fosse una funzione che segnalava la necessità di compilare tutti i campi prima di poter accedere alla prossima sezione, infatti quando durante i testi quando premiamo il tasto avanti esce un pop-up con indicando di compilare tutti i dati richiesti.