UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Informatica III

Modulo di Progettazione e Algoritmi

Relazione e documentazione sul progetto “ticket-generator”

Prof. Patrizia Scandurra

Salvatore Greco matr.1053509

Fabio Gamba matr. 1053157

Alessandro Chaar matr. 1054918

A.A 2022/2023

Indice

Introduzione al progetto

Il progetto scelto vuole implementare un sistema di generazione automatica dei biglietti per rispettare il proprio turno all'interno di edifici pubblici, come ad esempio i sistemi presenti negli uffici postali o bancari. In particolare questo sistema avrà una forte personalizzazione per un CUP (Centro Unico di Prenotazione) ospedaliero, rivolgendosi quindi al settore sanitario. Un punto chiave che lo differenza dal classico strumento "cartaceo", nonché motivo principale dello sviluppo, è la sua funzionalità IOT, ovvero di avere questo sistema connesso ad internet in modo che possa svolgere il servizio anche al di fuori dell'ambiente locale di utilizzo. Questa funzionalità è richiesta per una motivazione molto importante: quella di poter prenotare un turno senza essere fisicamente presenti per ritirare il biglietto stampato, così come la possibilità di prenotare giorno e ora esatta all'interno del sistema slegandosi dal concetto più antiquato di "coda di attesa".

Obiettivo principale del progetto è lo sviluppo del software utilizzando la metodologia AGILE, diminuendo costi e tempi di sviluppo rispetto alle metodologie standard

Toolchain

Implementazione del software

- Linguaggio di programmazione: Java
- IDE: IntellJ IDEA, (Android Studio), Eclipse
- DBMS: MongoDB Atlas
- GUI: Window Builder

Modellazione

- Use case diagram: app.diagrams.net (ex draw.io)
- Deployment diagram, component diagram, class diagram, sequence diagram, datatype datagram: Astah UML

Analisi del software

- Analisi statica: Stan4J
- Analisi dinamica: JUnit

Documentazione, versioning e organizzazione del team

- Documentazione: Microsoft Word
- Versioning: Git, GitHub come hosting
- Organizzazione team: GitHub, Discord

Organizzazione del team

Superate le prime iterazioni, il team ha usufruito di alcune funzionalità di GitHub, come le Issues e una logica di funzionamento dei branch, per scandire e parallelizzare il lavoro sulla base delle funzionalità da implementare, ad esempio assegnando diverse priorità ai task che vengono generati man mano che si procede nello sviluppo assegnando diverse priorità ai task che vengono generati man mano che si procede nello sviluppo

- Issues: consentono agli sviluppatori di capire immediatamente il lavoro che c'è da svolgere sul codice (anche con l'utilizzo di labels) per implementare le funzionalità o fixare comportamenti indesiderati. Uno sviluppatore può creare e assegnare issues ad altri collaboratori in modo da suddividere il lavoro sulla base delle risorse personali a disposizione (tempo, skills, esperienza…)
- Logica dei branch: ogni sviluppatore possiede il proprio branch derivato dal main branch. Su ogni branch personale avvengono le modifiche degli sviluppatori che poi vengono uniti al main branch tramite Pull Request
- Pull Request: si tiene traccia delle implementazioni fatte dagli sviluppatori nel main branch richiedendo la pull request
- Milestones: teniamo traccia delle funzionalità implementate a gruppi con le Milestones, che aiutano anche per il versioning del prodotto finale

Modello AGILE

Con questo metodo siamo riusciti a gestire variabili (come: tempo, risorse e qualità) in modo molto più efficiente che con altre metodologie. Nasce come bisogno di contrapporsi con i modelli tradizionali come il modello a spirale e il modello a cascata che sono spesso bloccanti in alcune fasi dello sviluppo. Questa nuova metodologia pensa il software in modo leggero e ci permette di raggiungere grande efficienza con l'utilizzo di best-practice. I quattro valori agili importanti sono:

- Persone e interazioni sono più importanti del processo e dei tools utilizzati.
- Un software funzionante è più importante di una chiara documentazione.
- La collaborazione del cliente va aldilà del contratto stipulato.
- La capacità di rispondere a cambiamenti è una caratteristica fondamentale. Può succedere spesso che vengano imposte delle modiche al prodotto: bisogna adottare un processo in grado di apportare continui cambiamenti a ciò che stiamo realizzando.

Casi di studio esempio: CoCoME e e-Campus

Per lo sviluppo di questo progetto abbiamo seguito come riferimento le tecniche e modelli visti a lezione con il caso di studio CoCoME e anche con il progetto e-Campus dei nostri colleghi. Il caso di studio e-Campus ha avuto un occhio di riguardo in più per quanto riguarda le practice utilizzate per la modellazione UML

Iterazione 0

0.1 Analisi dei requisiti

Abbiamo svolto questa fase di analisi cercando di immaginare tutti gli scenari possibili sia dal punto di vista del paziente, sia dal punto di vista dell'operatore ospedaliero. Per raggiungere questo obiettivo abbiamo usato la tecnica degli Use Case Stories, grazie alla quale abbiamo preso ogni attore separatamente e abbiamo individuato le funzionalità che l'applicazione dovrà avere.

0.2 Attori

L'analisi degli Use Case ha reso necessario individuare gli attori coinvolti nell'utilizzo del sistema. Gli attori identificati vengono divisi in attori primari e secondari e sono i seguenti:

- Utente standard (primario)
- Utente operatore che si occupa delle richieste degli utenti standard (primario)
- Sistema di visualizzazione della coda (secondario)
- Sistema di autenticazione (secondario)
- Sistema di visualizzazione (secondario)

0.3 Use case stories

0.3.1 Utente standard

Generazione biglietto

- Un utente che chiede il biglietto vuole avere il numero assegnato
- Un utente può voler scegliere tra code di più servizi
- Un utente vuole stampato il biglietto
- Un utente lo vuole visualizzato sullo schermo della propria app

Registrazione profilo utente

- Un utente può effettuare la registrazione con le proprie informazioni.

- Un utente può richiedere la generazione di una nuova password se ha smarrito quella attuale.

Visualizzazione della dashboard (coda)

- Un utente può voler visualizzare la coda su schermo
- Un utente vuole sapere qual è il suo turno guardando sullo schermo
- Un utente vuole sapere a che sportello è stato assegnato
- Un utente può voler visualizzare il tempo stimato per il suo turno

Gestione della coda (coda prioritaria)

- Un utente può voler generare un biglietto di coda prioritaria anche se non si trova nell'ufficio degli operatori
- Un utente può voler generare un biglietto di coda prioritaria per velocizzare il procedimento
- Un utente può voler guardare la coda attuale senza essere nell'ufficio

Login/logout

- Un utente può voler accedere tramite login per ottenere un biglietto nella coda prioritaria

Gestione profilo utente

- Un utente può voler visualizzare il tempo stimato per il suo turno
- Un utente può voler visualizzare una cronologia di prenotazioni generate all'interno del proprio profilo
- Un utente può voler visualizzare/modificare i suoi dati
- Un utente può richiedere una nuova password

0.3.2 Utente operatore

Gestione della coda

- Un operatore vuole ricevere il numero che deve servire quando passa al successivo
- Un operatore vuole che la coda visualizzata scorra quando seleziona il successivo
- Un operatore, per selezionare il successivo, schiaccia un pulsante

0.3.3 Sistema di gestione della coda

Gestione della coda

- Elabora l'avanzamento della coda
- Genera biglietti standard
- permette l'accesso alla funzionalità di coda prioritaria

0.3.4 Sistema di autenticazione

Gestione della coda

- Il sistema di autenticazione permette l'accesso alla funzionalità di coda prioritaria

Registrazione profilo utente

- Permette la registrazione solo agli operatori allo sportello

0.3.5 Sistema di visualizzazione

Visualizzazione della dashboard (coda)

- Deve visualizzare i biglietti serviti attualmente dagli operatori
- Deve visualizzare i biglietti immediatamente successivi in coda (alcuni)
- Deve visualizzare correttamente lo scorrere della coda

0.4 Use Case Diagram

Riportiamo tutti gli use case specificati nello schema per meglio capire le associazioni tra gli attori e l'inglobamento di varie funzionalità nei sotto-sistemi. Da questo schema si intravedono tutte le funzionalità del sistema complessivo

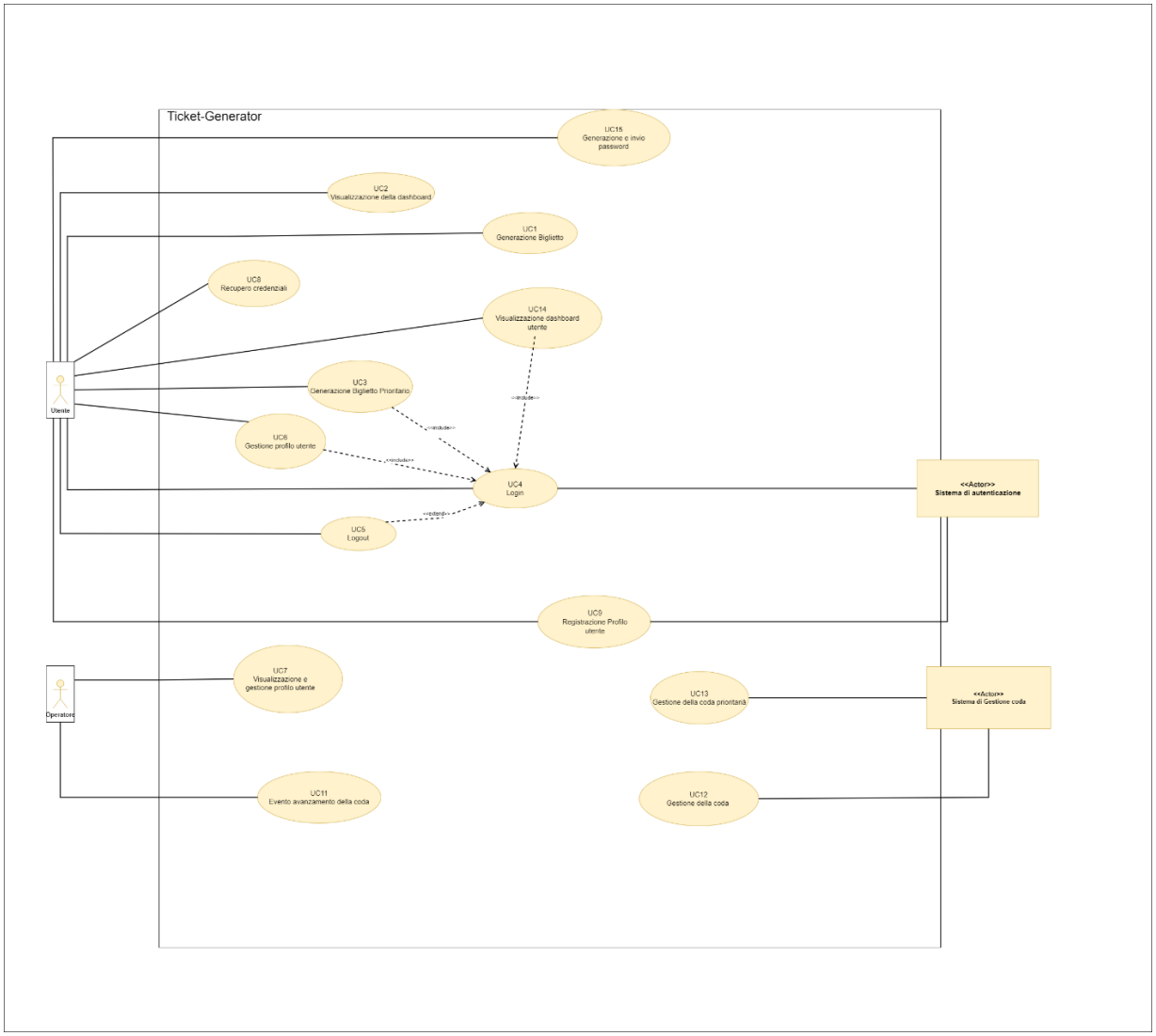

Figura 1: Use Case Diagram

0.5 Use Case Description (descrizione degli use case)

0.5.1 UC1: Generazione biglietto

UseCase	Generazione biglietto
Summary	Generazione del biglietto per l'utente
Actor	Utente
Precondition	L'utente non ha il biglietto
Postcondition	L'utente ha il biglietto
Base sequence	1. L'utente inizializza la procedura di generazione a schermo 2. Dopo aver selezionato il servizio, conferma la scelta 3. Il sistema stampa/visualizza a schermo il biglietto generato
Branch sequence	
Exception sequence	
Sub UseCase	
Note	

0.5.2 UC2: Visualizzazione della dashboard della coda

UseCase	Visualizzazione della dashboard (coda)
Summary	Visualizzazione delle informazioni della coda
Actor	Utente
Precondition	
Postcondition	
Base sequence	1. L'utente dopo aver ricevuto il biglietto attende il proprio turno 2. Il turno può avvenire quando a schermo viene visualizzato il biglietto con un operatore assegnato. In particolare, sulla dashboard vengono visualizzati per ogni riga - Biglietto - Operatore (se presente) - Tempo stimato
Branch sequence	
Exception sequence	
Sub UseCase	
Note	

0.5.3 UC3: Generazione biglietto prioritario

UseCase	Generazione Biglietto prioritario
Summary	Generazione di un biglietto prioritario
Actor	Utente
Precondition	L'utente ha fatto il login
Postcondition	
Base sequence	1. L'utente inizializza l'operazione di generazione da app 2. L'utente visualizza il tempo stimato 3. L'utente conferma la generazione, visualizzando a schermo il biglietto generato
Branch sequence	
Exception sequence	
Sub UseCase	Log-in
Note	

0.5.4 UC4: Log-in

UseCase	Log-in
Summary	Accesso al sistema di priorità
Actor	Utente
Precondition	L'utente è registrato
Postcondition	
Base sequence	1. L'utente inizializza la procedura di accesso inserendo i propri dati di autenticazione 2. L'utente visualizza a schermo la riuscita dell'operazione
Branch sequence	
Exception sequence	1. Viene generato un errore di autenticazione poiché l'utente o non è registrato o ha sbagliato l'inserimento dei dati 2. L'utente visualizza a schermo la nuova richiesta di dati di autenticazione
Sub UseCase	
Note	

0.5.5 UC5: Log-out

UseCase	Log-out
Summary	Uscita dal sistema di priorità
Actor	Utente
Precondition	L'utente è loggato
Postcondition	
Base sequence	1. L'utente inizializza la procedura schiacciando il tasto di logout 2. L'utente viene riportato alla pagina di autenticazione
Branch sequence	
Exception sequence	
Sub UseCase	
Note	

0.5.6 UC6: Gestione profilo utente

UseCase	Gestione profilo utente
Summary	Modifica e visualizzazione dei dati utente
Actor	Utente
Precondition	L'utente ha fatto il login
Postcondition	
Base sequence	1. L'utente clicca sul pulsante adibito alla visualizzazione del proprio profilo 2. La schermata mostra le informazioni inserite in fase di registrazione con l'operatore Viene mostrato un pulsante per abilitare la modifica del form, oppure tornare indietro 3. L'utente alla fine torna alla schermata principale
Branch sequence	B1: 3. L'utente ha cliccato sulla modifica del form abilitandola 4. L'utente dopo aver compilato il form può decidere se confermare le modifiche o tornare alla schermata di visualizzazione
Exception sequence	
Sub UseCase	
Note	

0.5.7 UC7: Gestione profili utenti

UseCase	Visualizzazione e gestione profilo utenti
Summary	Visualizzazione e modifica degli utenti registrati
Actor	Operatore
Precondition	L'operatore è seduto alla propria postazione
Postcondition	
Base sequence	1. L'operatore clicca sul pulsante adibito alla visualizzazione degli utenti registrati 2. La schermata mostra la lista degli utenti con una barra di ricerca 3. Selezionato l'utente desiderato, viene visualizzato e resa possibile la modifica del form 4. L'operatore alla fine torna alla schermata principale
Branch sequence	
Exception sequence	
Sub UseCase	
Note	

0.5.8 UC8: Registrazione profilo utente

UseCase	Registrazione profilo utente
Summary	Per l'accesso al sistema di priorità da parte dell'utente
Actor	Utente
Precondition	L'utente ha richiesto la registrazione
Postcondition	
Base sequence	1. Un utente richiede la registrazione tramite app 2. L'utente compila un apposito form 3. L'utente aspetta la ricezione della password tramite SMS. [E1: utente già registrato] [E2: messaggio non ricevuto]
Branch sequence	
Exception sequence	E1: 4. L'utente è già registrato 5. Viene annullata la registrazione E2: 4. Messaggio non ricevuto 5.
Sub UseCase	
Note	

0.5.9 UC9: Registrazione profilo utente

UseCase	Registrazione profilo utente
Summary	
Actor	Sistema di autenticazione
Precondition	L'utente ha compilato il form
Postcondition	Il sistema invia la password all'utente via SMS
Base sequence	1. Il sistema controlla la validità del form [E1: dati non validi] [E2: utente già registrato] 2. Il sistema registra l'utente e le sue informazioni 3. Il sistema invia la password generata tramite SMS al numero specificato entro 10 minuti 4. Il sistema completa la registrazione al completamento del primo accesso da parte dell'utente [E3: l'utente non esegue l'accesso]
Branch sequence	B1: 4. Viene mostrato l'utente trovato in una nuova pagina
Exception sequence	E1: 2. I dati da inserire non sono validi 3. Il sistema restituisce un errore concorde E2: 2. L'utente è già registrato 3. Il sistema chiede se vogliamo passare alla scheda dell'utente trovato [B1: Gestione profilo utenti] E3: 4. Il sistema cancella l'utente che non ha effettuato il primo accesso entro 24 ore
Sub UseCase	Registrazione profilo utente
Note	

0.5.10 UC10: Recupero Credenziali

UseCase	Recupero Credenziali
Summary	Alla richiesta dell'utente si può inviare una nuova password
Actor	Utente
Precondition	
Postcondition	
Base sequence	1. L'utente clicca sul pulsante adibito all'invio di una nuova password
Branch sequence	
Exception sequence	
Sub UseCase	Gestione del profilo utenti
Note	

0.5.11 UC11: Evento di avanzamento della coda

UseCase	Evento di avanzamento della coda
Summary	Funzionalità all'interno del sistema di gestione della coda disponibile per farla avanzare
Actor	Operatore
Precondition	L'operatore ha schiacciato
Postcondition	La coda è avanzata
Base sequence	1. L'operatore segnala la propria disponibilità schiacciando il pulsante 2. L'operatore riceve il biglietto da servire e viene visualizzata a schermo l'assegnazione all'operatore [E1: coda vuota] 3. Dalla visualizzazione della coda viene cancellato il biglietto che stava servendo
Branch sequence	
Exception sequence	E1: 2. Non c'è nessuno in coda (non ci sono utenti che hanno richiesto un biglietto), l'operatore riceve un messaggio di default
Sub UseCase	
Note	

0.5.12 UC12: Gestione della coda

UseCase	Gestione della coda
Summary	Funzionalità del sistema di gestione della coda su evento dell'operatore
Actor	Sistema di gestione della coda
Precondition	
Postcondition	La coda è avanzata
Base sequence	1. Il sistema riceve il numero operatore successivamente alla pressione del pulsante da parte di esso 2. A questo punto controlla la coda, eliminando un biglietto associato se presente 3. Assegna il primo biglietto disponibile (non ha un operatore associato) all'operatore della richiesta
Branch sequence	
Exception sequence	E1: 3. Non c'è nessuno in coda (non ci sono utenti che hanno richiesto un biglietto), l'operatore riceve un messaggio di default
Sub UseCase	Evento di avanzamento della coda
Note	

0.5.13 UC13: Gestione della coda prioritaria

UseCase	Gestione della coda prioritaria
Summary	Funzionalità per l'utente di generare un biglietto prioritario
Actor	Sistema di gestione della coda
Precondition	L'utente ha fatto l'accesso
Postcondition	L'utente ha il biglietto
Base sequence	1. Il sistema genera il biglietto con nomenclatura speciale 2. Il sistema invia il biglietto all'utente richiedente
Branch sequence	
Exception sequence	
Sub UseCase	
Note	

0.5.14 UC14: Visualizzazione dashboard utente

UseCase	Visualizzazione dashboard utente
Summary	Visualizzazione delle prenotazioni, visualizzazione tempo stimato
Actor	Utente
Precondition	L'utente ha fatto il login
Postcondition	
Base sequence	1. L'utente clicca sul pulsante adibito alla visualizzazione delle proprie prenotazioni 2. La schermata mostra le prenotazioni effettuate come una lista. Le prenotazioni ancora valide mostreranno un tempo stimato, mentre quelle passate no 3. L'utente alla fine torna alla schermata principale
Branch sequence	
Exception sequence	
Sub UseCase	
Note	

0.5.15 UC15: Generazione e invio password

UseCase	Generazione e invio password
Summary	Alla richiesta dell'utente si può inviare una nuova password
Actor	Utente
Precondition	L'utente ha aperto l'applicazione
Postcondition	
Base sequence	1. L'utente clicca sulla richiesta di generazione nuova password 2. L'utente inserisce il proprio numero di telefono [E1: il numero di telefono non è registrato] 3. L'utente riceve una nuova password via SMS
Branch sequence	
Exception sequence	E1: 3. Viene restituito un messaggio di errore
Sub UseCase	Gestione del profilo utenti da parte dell'operatore
Note	

0.6 Analisi delle specifiche

0.6.1 Introduzione

Le specifiche descritte sono state ordinate in modo da implementare prima le funzionalità necessarie (e quindi vincoli) alle successive funzionalità del software. Sono quindi divise in specifiche con priorità alta e bassa.

0.6.2 Specifiche funzionali

- R1 Generazione biglietto in sede: l'applicazione possiede la capacità di generare un biglietto per l'utente in sede nell'ufficio. Implica che il biglietto venga fornito e visualizzato dall'utente e inserito nella lista d'attesa del sistema in base anche al tipo richiesto al momento della generazione
- R2 Visualizzazione coda: nella sede è presente un device adibito alla visualizzazione della coda generata. Il sistema sviluppato ha bisogno quindi di questa funzionalità da utilizzare nel device per la visualizzazione agi utenti in coda che hanno richiesto e ottenuto il biglietto
- R3 Avanzamento coda (operatore): il sistema deve poter scalare su richiesta dell'operatore, che richiederà l'operazione quando termina il servizio con l'utente precedentemente servito. Quindi il sistema seleziona il prossimo biglietto da servire in base alla logica progettata
- R4 Profilazione utente: il sistema tiene traccia degli utenti se richiesto, tramite un sistema di profilazione affidato all'operatore. La profilazione viene poi utilizzata per le successive funzionalità
- R5 Log in/Log out utente al sistema online: l'applicazione permette l'accesso esclusivo all'utente che lo richiede
- R6 Generazione biglietto non in sede: l'applicazione permette di generare e tenere traccia dei biglietti tramite applicazione che comunica con il sistema in sede, così da permettere una generazione non in sede

Codice	Nome	Priorità	Implementato
R1	Generazione biglietto in sede	A	Si
R2	Visualizzazione coda	A	Ni
R3	Avanzamento coda (operatore)	A	Si
R4	Profilazione utente	B	No
R5	Log in/Log out	B	No
R6	Generazione biglietto non in sede	B	No

Tabella 1: Requisiti funzionali

0.6.3 Logica di funzionamento delle code

In particolare, definiamo la business logic dietro il sistema di espletamento della coda, che avviene alla pressione da parte dell'operatore del pulsante (segnalazione di disponibilità).

Questa funzionalità viene solo abbozzata ma iniziamo a considerare un buon funzionamento per questa funzionalità di sistema fondamentale

- Accettazione (di una visita già prenotata) -> priorità/biglietto A
- Priorità o biglietto online -> priorità/biglietto X
- Prenotazione (con ricetta oppure senza) -> priorità/biglietto B
- Ritiro referti -> priorità/biglietto C1
- Registrazione nel sistema di login priorità -> priorità/biglietto C2

Su 10 serviti, 4 sono accettazione, 3 sono priorità, 2 sono prenotazione, 1 sono in coda C.

Il sistema dovrà quindi ripartire i serviti sulla base di questa suddivisione

0.7 Analisi dell’architettura

Abbiamo realizzato due deployment diagram: il primo serve per avere un’idea generale dell’intero sistema, mentre il secondo per mostrare nel dettaglio la struttura delle componenti software su quelle hardware.

0.7.1 Deployment diagram - Architettura

Questo deployment diagram mostra i dispositivi che vengono utilizzati e come interagiscono fra di loro. L'architettura è formata da un server centrale che interagisce con un database e con le seguenti componenti hardware:

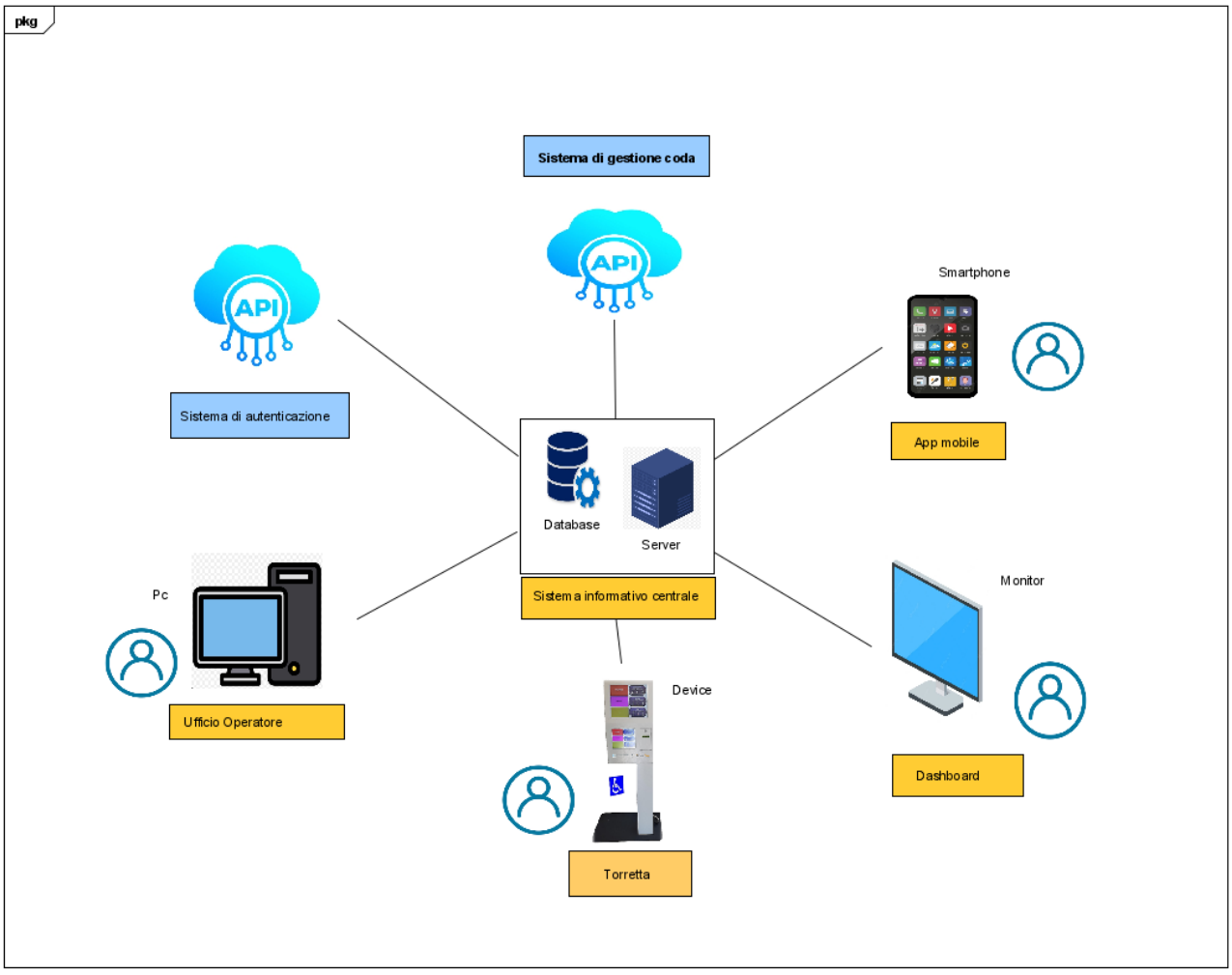

Figura 2: Deployment diagram informale

- Ufficio operatore, dove l’utente può richiedere la procedura di registrazione
- Dashboard, per vedere la coda attuale
- App mobile, per prenotare il biglietto nella coda prioritaria e per vedere le informazioni personali.
- Sistema di gestione coda, che gestisce l’avanzamento della coda.
- Sistema di autenticazione, che gestisce l'autenticazione degli utenti.

0.7.2 Deployment diagram – UML

Questo diagram approfondisce le informazioni contenute nel diagram precedente, specificando la tecnologia di comunicazione utilizzata.

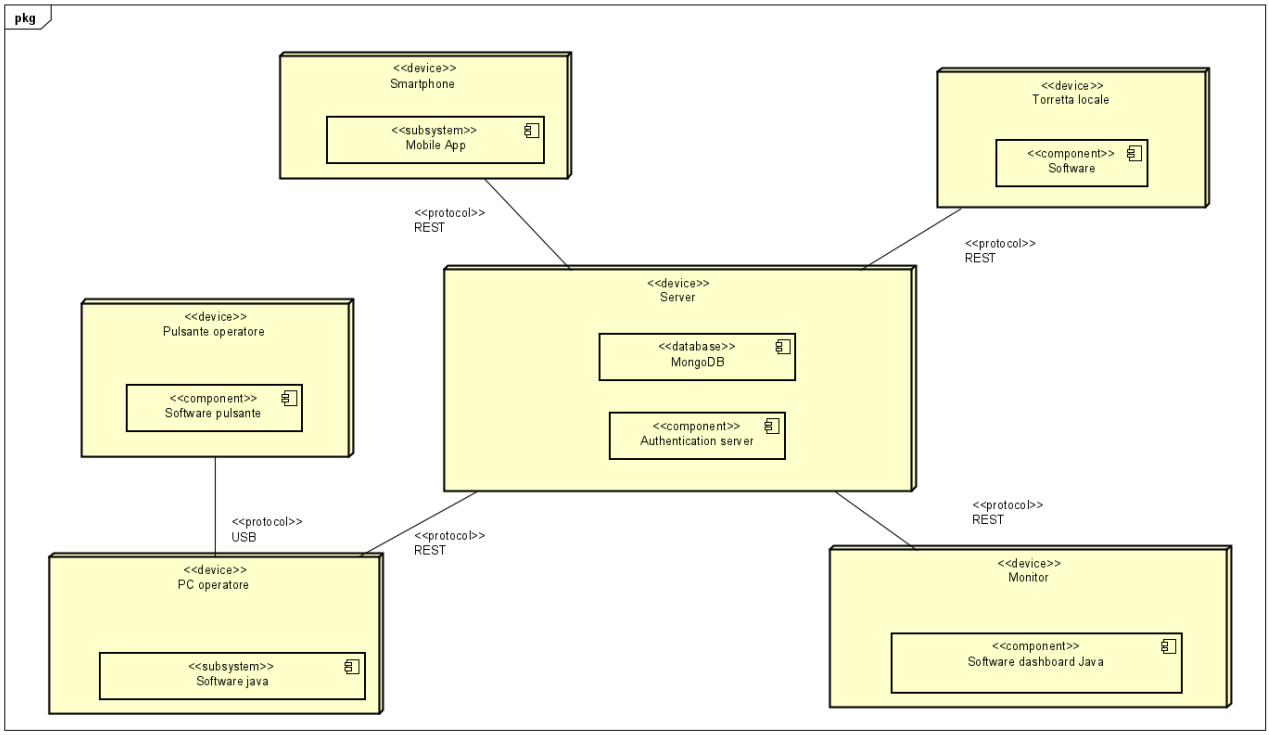

Figura 3: Deployment diagram UML

Iterazione 1

1.1 Component diagram – interfacce

Il diagramma delle interfacce mostra come i componenti del sistema comunicano ed interagiscono tra di loro per implementare le funzionalità richieste dall'analisi dei requisiti.

Le interfacce definite non sono definitive (presentano infatti un nome abbozzato) ma semplicemente prototipate per adattarsi poi durante lo sviluppo e le iterazioni successive del processo agile.

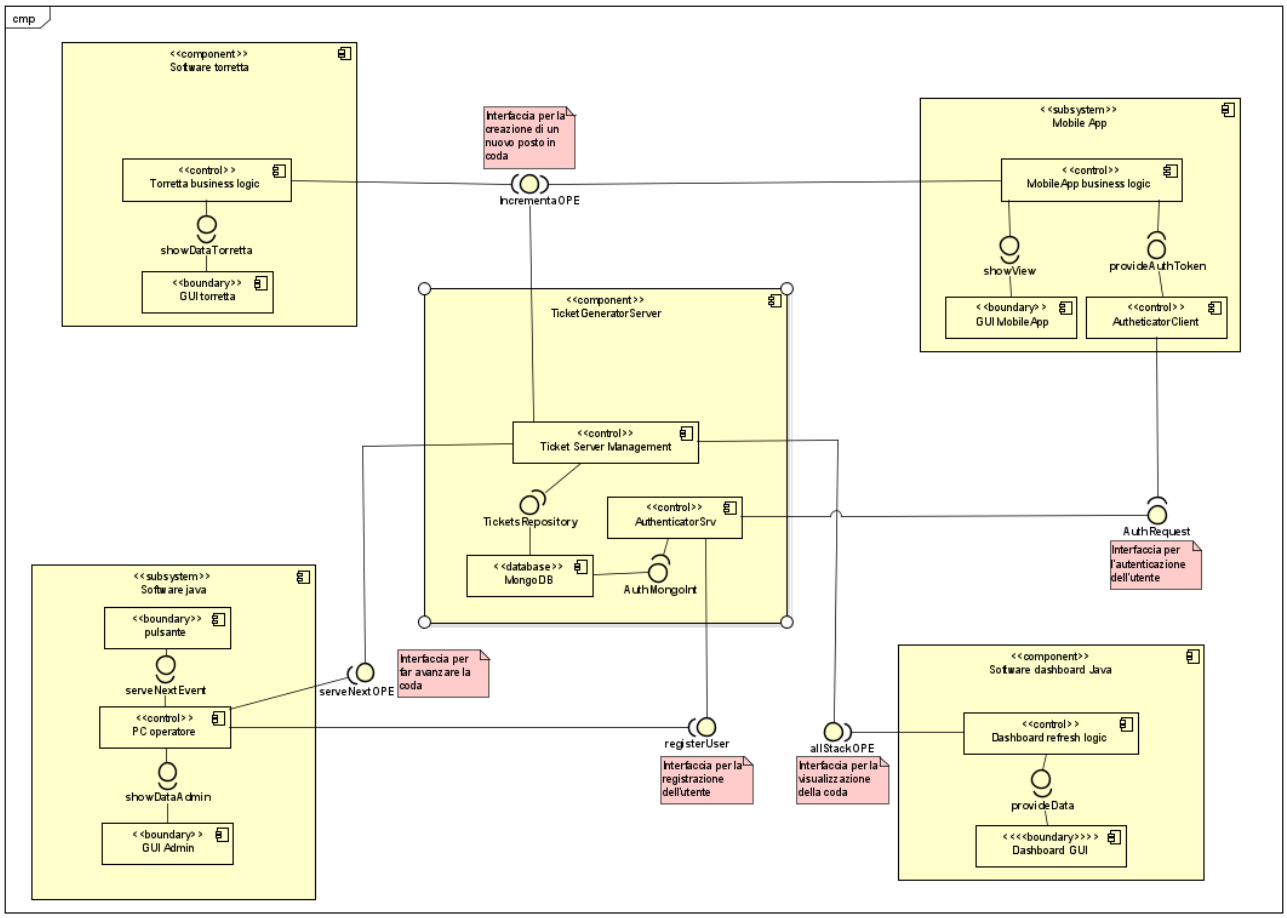

Figura 4: Component diagram delle interfacce

1.2 Class diagram - Interface definition

Le interfacce abbozzate del component diagram vengono definite più nello specifico in questo diagramma, dove specifichiamo per ogni interfaccia i metodi necessari per implementare la funzionalità tra i componenti che la utilizzano

Anche qui le firme dei metodi non sono completamente definite, in modo che vengano specificate al bisogno durante le iterazioni successive, nel processo di sviluppo del software

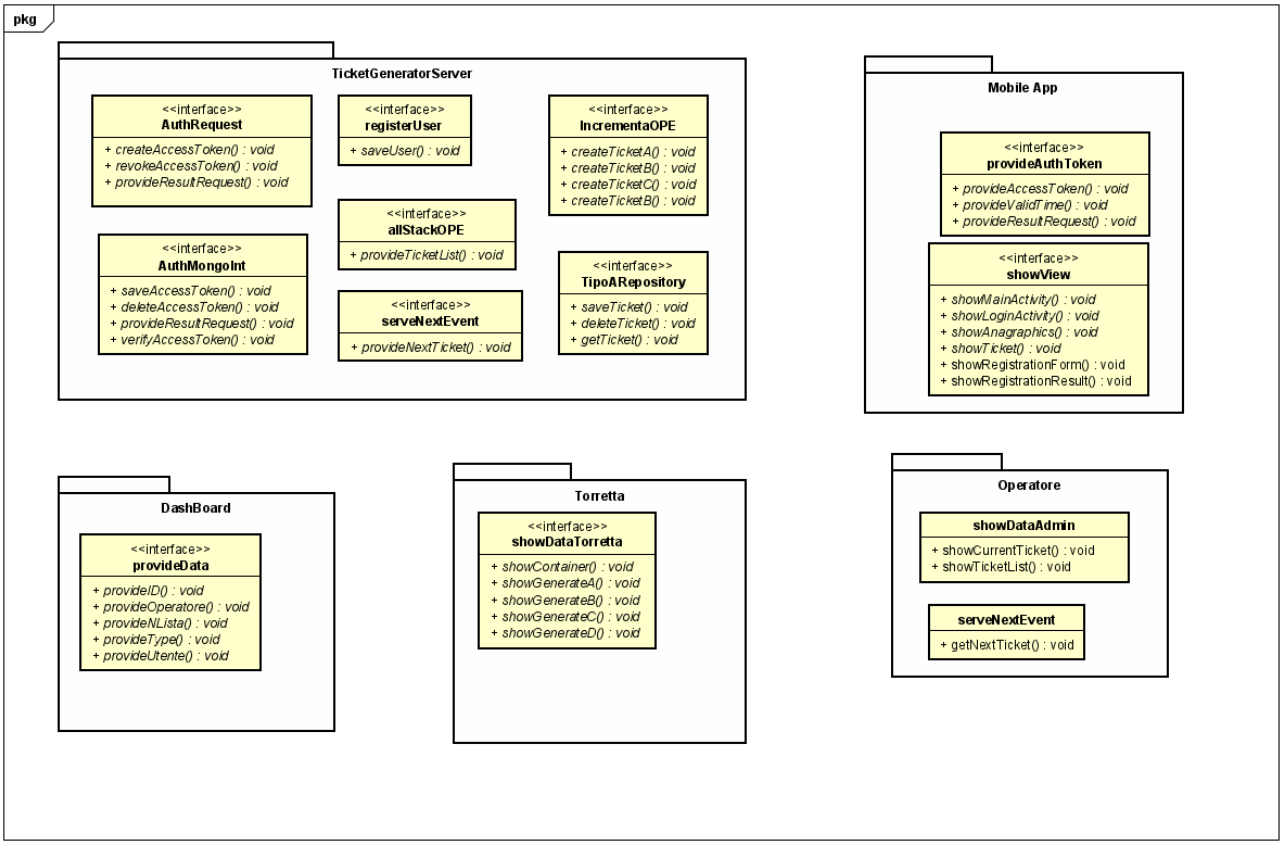

Figura 5: Class diagram, interface definition

1.3 Class diagram - Data type (presentation model)

Rappresenta le relazioni statiche tra gli oggetti che contengono informazioni, cosi come una possibile struttura della basi di dati.

Abbiamo utilizzato una particolare struttura di ereditarietà ed interfacce perché nelle iterazioni successive vogliamo, con un unico oggetto, scambiare sia l'input di un componente che l'output, in base all'API che verrà chiamata. Il principio è che un componente mittente crea l'oggetto e ne popola l'input, mentre un componente destinatario lo riceve e ne popola l'output, restituendolo al mittente.

In questo modo possiamo facilmente tenere traccia delle richieste all'interno del sistema e monitorarne la logica di funzionamento

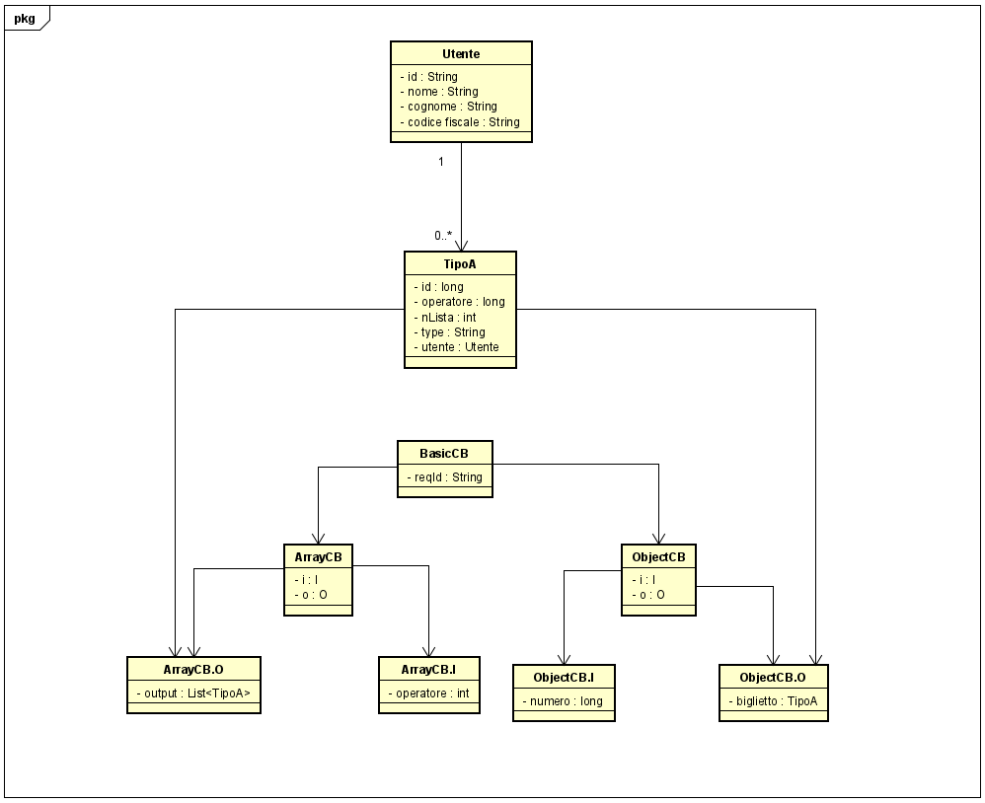

Figura 6: Class diagram, data type

Iterazione 2

2.1 Tecnologie utilizzate

Una volta decisa l'architettura di base nelle fasi iniziali, abbiamo pensato alle migliori tecnologie per implementarne il funzionamento, con un occhio di riguardo verso le novità del momento

2.1.1 Database - MongoDB

La decisione è ricaduta su un database NoSQL per via dell'estrema libertà di utilizzo che consente, un DBMS non relazionale che facilita la memorizzazione degli oggetti Json del backend. L'hosting è a cura di MongoDB Atlas, anche se in fase di deployment è anche possibile utilizzare una macchina con Docker Desktop con un'immagine di MongoDB

2.1.2 Framework Spring

Il framework Spring è un framework open source per lo sviluppo di applicativi Java, in particolare per quanto riguarda la parte backend. All'interno del nostro progetto utilizziamo maggiormente 2 delle sue componenti

- Spring MVC: per l'implementazione delle API REST esposte dalla parte server, utilizzate sia per la componente di autenticazione, sia per la parte di funzionalità
- Spring Data: mette a disposizione dei driver già implementati per la comunicazione con i DBMS. Abbiamo utilizzato questa funzionalità come driver di comunicazione con MongoDB Atlas

2.1.3 Eclipse IDE – Window Builder

Abbiamo utilizzato Eclipse come IDE in particolare per lo sviluppo dei client, soprattutto la parte di GUI. Eclipse mette a disposizione lo strumento Window Builder che aiuta lo sviluppatore a costruire un'interfaccia per applicativi Java, utilizzando le librerie integrate Java Swing

2.1.4 Repository - GitHub

La tecnologia impiegata per la repository è Git, su hosting GitHub. Questo strumento è fondamentale sia per il versioning del progetto, sia per parallelizzare il lavoro su più collaboratori. In particolare abbiamo utilizzato alcune delle funzionalità di GutHub per scandire la fase di sviluppo del software, come illustrato più avanti

2.1.5 Testing - Postman

Per il testing a lavori in corso delle API esposte dal backend abbiamo utilizzato il software Postman, che si occupa di "mockare" le richieste inviate dai client verso il server e di riceverne la risposta. In questo modo abbiamo velocizzato lo sviluppo dell'elemento portante del progetto, testando la parte backend, senza avere a disposizione i client che sono stati poi successivamente "cuciti" sulle API del backend.