



Documentazione progetto del

Modulo di Progettazione, Algoritmi e Computabilità

cod. corso 38068

Laurea Magistrale in Ingegneria Informatica

Università degli Studi di Bergamo

A.A. 2023/204

Prof.ssa Patrizia Scandurra

Componenti del gruppo:

Davide Gamba, matr. 1053470

Giorgio Tentori, matr. 1053248

Aurora Zanenga, matr. 1054891

INTRODUZIONE AL PROGETTO	1
TECNOLOGIE E METODI.....	2
AGILE MODEL DRIVEN DEVELOPMENT AMDD	2
TOOLCHAIN	3
1. ITERAZIONE 0.....	7
1.1 ANALISI DEI REQUISITI	7
1.2 USE CASE DIAGRAM	8
1.3 ANALISI DELLE SPECIFICHE	9
1.4 ANALISI DELL'ARCHITETTURA	12
1.5 DEPLOYMENT DIAGRAM	13
2. ITERAZIONE 1.....	14
2.1 INTRODUZIONE.....	14
2.2 UC1 – VISUALIZZAZIONE HOMEPAGE.....	14
2.3 UC2 – VISUALIZZAZIONE BACHECA	15
2.4 UC3 – REGISTRAZIONE UTENTE	17
2.5 UC4 – LOGIN UTENTE	18
2.6. UC5 – LOGOUT UTENTE	19
2.7 - COMPONENT DIAGRAM.....	20
2.8 - UML CLASS DIAGRAM PER LE INTERFACCE	22
2.9 - UML CLASS DIAGRAM PER I TIPI DI DATO	24
2.10 – TESTING	26
3. ITERAZIONE 2.....	32
3.1 INTRODUZIONE.....	32
3.2 UC6 – ACQUISTO DI UN PRODOTTO.....	32
3.3. UC7 - SVILUPPO E VISUALIZZAZIONE ALGORITMO DI RANKING.....	33
3.4 - ALGORITMO DI RANKING.....	34
3.5 - UML COMPONENT DIAGRAM	39
3.6 - UML CLASS DIAGRAM PER LE INTERFACCE	40
3.7 - UML CLASS DIAGRAM PER I TIPI DI DATO	41
3.8 – TESTING	42

3.9 – SCREENSHOT APPLICAZIONE.....	46
------------------------------------	----

Introduzione al progetto

L'obiettivo primario di questo progetto consiste nella creazione di un sito di e-commerce dedicato alla compravendita online di notebook.

L'applicazione sviluppata consentirà agli utenti di esplorare una bacheca contenente i prodotti disponibili, effettuare ricerche personalizzate in base alle specifiche desiderate per i notebook e, qualora desiderino acquistare o vendere tali dispositivi, sarà necessario registrarsi sulla piattaforma.

La finalità del progetto è fornire agli utenti la possibilità di mettere in vendita i propri notebook non utilizzati e, al contempo, di esplorare le offerte disponibili per i loro prossimi acquisti.

Al fine di garantire l'affidabilità degli utenti, l'applicazione incorporerà la funzionalità di valutazione del venditore dopo ogni transazione.

L'interfaccia dell'applicazione si aprirà con una home page di presentazione, dalla quale gli utenti potranno navigare agevolmente attraverso le varie sezioni del sito.

Tra le opzioni disponibili, si includono la visualizzazione dei prodotti nella bacheca, la registrazione o l'accesso alla propria pagina personale mediante login, la consultazione delle schede prodotto, la gestione del carrello degli acquisti e la conclusione degli ordini.

Per fornire una panoramica più dettagliata delle funzionalità del sistema informativo, è stato elaborato uno "Use case diagram" che sarà presentato nei paragrafi successivi.

Tecnologie e metodi

Agile Model Driven Development AMDD

Il modello di sviluppo adottato per questo progetto è l'Agile Model Driven Development (AMDD), un approccio che coniuga la flessibilità delle metodologie agili con la precisione derivante dallo sviluppo basato su modelli.

La coniugazione di questi elementi ha lo scopo di fornire un equilibrio tra capacità di adattamento alle evoluzioni dei requisiti (tipica delle metodologie agili) e chiarezza e la precisione derivanti dalla progettazione basata su modelli.

Le metodologie agili permettono di affrontare con agilità e adattabilità i cambiamenti continui dei requisiti durante lo sviluppo del software.

Dall'altra parte, lo sviluppo basato su modelli sottolinea l'importanza della progettazione come strumento per creare sistemi software complessi, offrendo un livello elevato di precisione.

Il processo di sviluppo inizia con la creazione di un modello ad alto livello all'inizio del progetto, noto come Iterazione 0.

Questo modello fornisce una visione chiara e completa del sistema che si intende sviluppare.

Nel corso delle iterazioni successive il modello viene costantemente raffinato e aggiornato parallelamente alla scrittura del codice. Ciò consente al modello di adattarsi in modo dinamico ai cambiamenti dei requisiti.

Le iterazioni consentono di mantenere un elevato grado di flessibilità e adattabilità, aspetti fondamentali quando si affrontano progetti complessi.

Toolchain

La toolchain rappresenta un insieme coordinato di software, strumenti e processi utilizzati nel ciclo di vita dello sviluppo software.

Per la realizzazione del progetto sono state utilizzate le seguenti tecnologie:

Modellazione del software

La modellazione del software è stata condotta seguendo il framework della "**4+1 View Model**".

Questo approccio fornisce una prospettiva completa sulla struttura e il comportamento di un sistema software attraverso l'elaborazione di **quattro viste principali**, integrate da una quinta vista che unisce e coordina le prospettive precedenti.

Nel contesto di questo progetto, sono state create le seguenti viste:

Use Case Diagram: Rappresenta interazioni e relazioni tra attori e casi d'uso, offrendo una panoramica chiara delle funzionalità offerte dal sistema.

Deployment Diagram: Illustra la disposizione fisica dei componenti del sistema, evidenziando la distribuzione su hardware o infrastruttura di rete.

Component Diagram: Fornisce una visione dettagliata della struttura interna del sistema, evidenziando i componenti software e le loro interazioni.

Class Diagram: Descrive la struttura statica del sistema, evidenziando classi, attributi e relazioni tra di esse.

Sequence Diagram: Offre una rappresentazione dinamica delle interazioni tra oggetti e classi nel sistema, mostrando la sequenza temporale delle operazioni.

L'adozione della "4+1 View Model" garantisce una comprensione approfondita e completa del software, facilitando la comunicazione e guidando l'implementazione e il testing del sistema.

Implementazione del Software

Per la fase di sviluppo del codice, sono stati adottati diversi strumenti che hanno contribuito a garantire efficienza e qualità nel processo di creazione del software:

- **Front-end: ReactJS**, un framework JavaScript utilizzato per la creazione di interfacce utente dinamiche e reattive.
- **Back-end: Spring Boot**, un framework Java che agevola lo sviluppo di applicazioni robuste e scalabili.
- **IDE (ambiente di sviluppo): Visual Studio Code**, una piattaforma che offre funzionalità avanzate per la scrittura del codice, il debugging e la gestione del progetto.
- **API Testing: Insomnia**, strumento utilizzato per verificare e validare le chiamate alle interfacce di programmazione.
- **Database: PostgreSQL**, un sistema di gestione di database relazionali noto per la sua affidabilità e flessibilità.
- **Versioning: GitHub**, una piattaforma che agevola la collaborazione tra sviluppatori e offre un sistema di controllo delle versioni distribuito.

L'integrazione di questi strumenti ha fornito una solida base per lo sviluppo del software, garantendo efficienza nel workflow e consentendo una gestione coerente e tracciabile delle varie fasi del progetto.

Analisi del codice

Nel contesto dell'analisi del codice, sono stati impiegati i seguenti strumenti:

- **Analisi Statica: RedHat**
Questo strumento offre un approccio completo all'analisi statica del codice, permettendo la valutazione approfondita del software senza l'esecuzione del programma. L'utilizzo di RedHat contribuisce alla rilevazione precoce di potenziali problematiche e all'assicurazione di standard di qualità elevati durante lo sviluppo.

- **Analisi Dinamica: JUnit**

JUnit è un framework di test per Java. Nella fase di analisi dinamica JUnit consente di verificare il comportamento e la correttezza delle singole unità di codice in un ambiente di esecuzione controllato.

Questo strumento è essenziale per la validazione e la verifica delle funzionalità del software in modo sistematico e automatizzato.

L'utilizzo combinato di analisi statica e dinamica contribuisce a garantire la qualità del codice, identificando potenziali errori o inefficienze durante lo sviluppo e facilitando la creazione di un software robusto e affidabile.

Organizzazione

La modellazione e la realizzazione del software sono state condotte attraverso sessioni di lavoro di gruppo, sia in modalità frontale sia da remoto.

Per la creazione dei diagrammi UML, è stato utilizzato **StarUML**, un tool che ha facilitato la rappresentazione e la comprensione della struttura e delle interazioni del sistema.

Per la documentazione, abbiamo utilizzato Microsoft Word, garantendo un ambiente familiare e integrato per la stesura e la condivisione dei documenti.

Per agevolare la condivisione del codice all'interno del nostro team, abbiamo adottato GitHub come piattaforma di gestione del versionamento.

In un contesto **agile** come quello seguito nel progetto, l'organizzazione del lavoro può avvenire seguendo i principi chiave dell'Agile Model Driven Development (AMDD).

La modalità di suddivisione del lavoro aveva la seguente base:

Pianificazione di Iterazioni: Sono state definite sotto-iterazioni temporali brevi in modo tale da potersi focalizzare su obiettivi specifici di sviluppo.

Assegnazione di Compiti Specifici: All'inizio di ogni iterazione sono stati assegnati dei compiti specifici a ciascun membro del team in base alle proprie competenze e aree di interesse.

Collaborazione Continua: Nonostante ciascuno avesse dei compiti specifici tra una sotto-iterazione e quella successiva sono state fatte regolarmente sessioni di confronto e aggiornamento sullo stato di avanzamento del lavoro di ciascuno.

La scelta di adottare la modalità agile ha notevolmente facilitato la collaborazione e la reciproca assistenza all'interno del nostro team. Ogni membro aveva una chiara comprensione di cosa stesse sviluppando il collega, consentendo un agevole scambio di conoscenze.

In caso di eventuali problematiche, il contesto agile ha permesso ai membri del gruppo di contribuire alla risoluzione, sfruttando la familiarità con il lavoro degli altri membri.

Raffinamento Continuo del Modello: In linea con i principi dell'AMDD, il modello iniziale è stato continuamente raffinato e migliorato durante le iterazioni successive.

Ogni membro del team ha contribuito all'evoluzione del modello in modo collaborativo.

1. Iterazione 0

1.1 Analisi dei Requisiti

Per condurre un'analisi completa dei requisiti, sono stati esaminati tutti i possibili scenari di utilizzo dell'applicazione, garantendo così una progettazione mirata e inclusiva.

Le principali funzionalità dell'applicazione sono state suddivise in diverse aree, considerando l'esperienza sia per gli utenti non registrati che per quelli registrati:

Home Page e Bacheca Prodotti: Gli utenti non registrati possono accedere al sito con funzionalità limitate, esplorando la home page, consultando i prodotti nella bacheca e le relative schede prodotto. L'opzione di registrazione è disponibile per accedere a tutte le funzionalità.

Area Personale: Dopo la registrazione, gli utenti hanno accesso completo all'applicazione. Possono effettuare il login e gestire la propria area personale, qui saranno riepilogati i loro dati (nome, cognome, mail, indirizzo, metodo per i pagamenti), gli acquisti effettuati (questi si dividono in ordini completati, ordini in corso e resi) ed i prodotti che ha messo in vendita (venduti oppure ancora sul mercato).

Ordini e Valutazioni: Gli utenti registrati possono acquistare i prodotti visibili in bacheca e/o pubblicare annunci di nuovi notebook in vendita, specificandone dettagli tecnici, prezzo e anni di utilizzo.

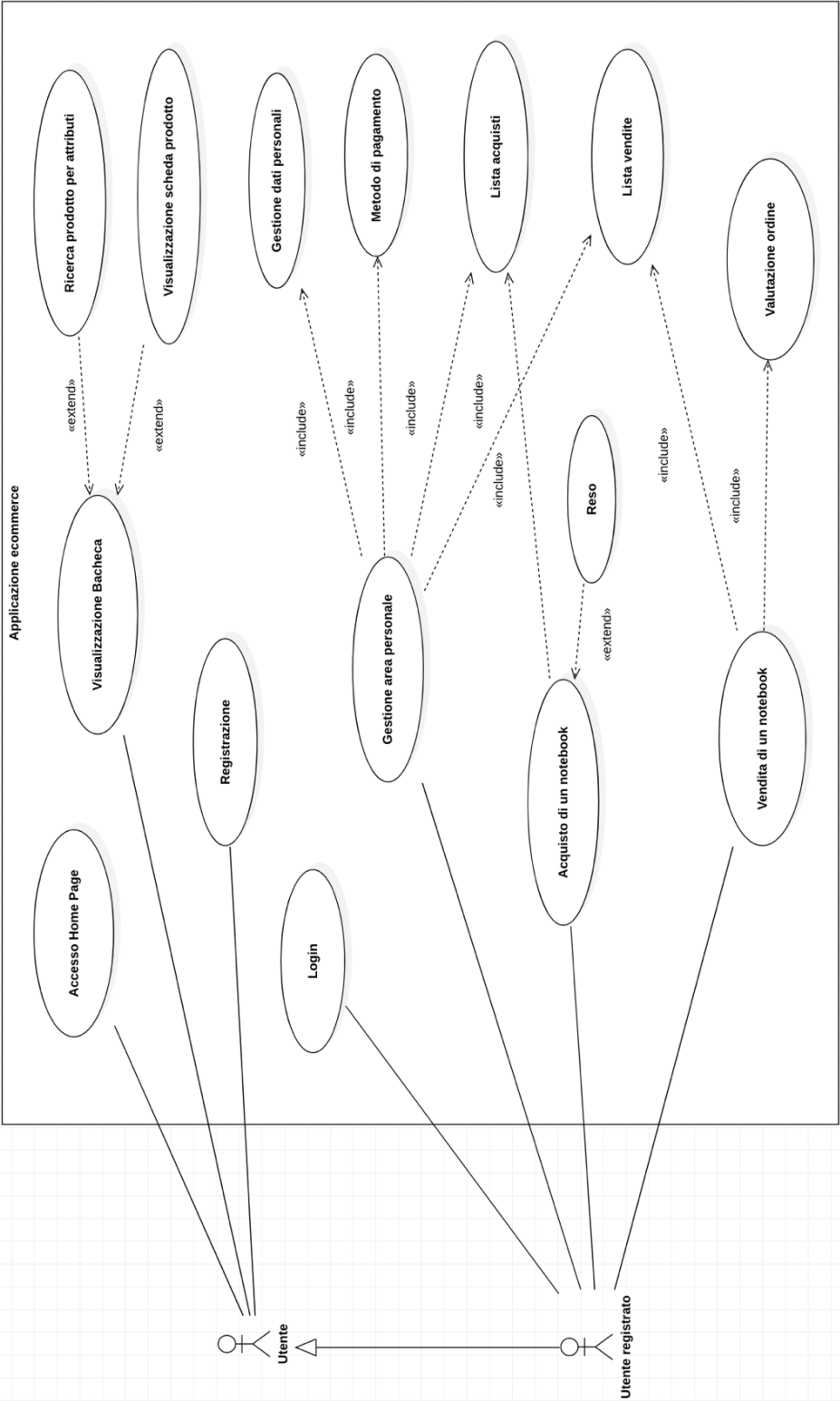
Dopo ogni acquisto, l'utente acquirente può rilasciare recensioni e assegnare valutazioni (da 1 a 5) all'utente venditore. Dalla pagina personale, è possibile effettuare resi entro 30 giorni dall'ordine.

Algoritmo di Ranking: L'applicazione implementa un significativo algoritmo di ranking che, assegnando una valutazione a ogni utente venditore al termine di ogni ordine, calcola un ranking basato sulla media di tutte le valutazioni.

L'algoritmo premia i venditori con punteggi più alti, rendendo più visibili i loro prodotti e presentando una classifica dei venditori.

Valutazione Venditore: Nel caso in cui un utente utilizzi la piattaforma solo per fare acquisti, la sua valutazione venditore sarà "UNRATED".

1.2 Use Case Diagram



1.3 Analisi delle specifiche

Ogni specifica del progetto è stata valutata e classificata in base alla sua priorità, considerando la sua urgenza per il corretto funzionamento del sistema.

Questa categorizzazione ha consentito di stabilire una roadmap di sviluppo, focalizzando gli sforzi sugli aspetti critici del progetto. Le specifiche sono state suddivise nelle seguenti priorità:

Alta Priorità: Queste specifiche rappresentano le funzionalità fondamentali per il sistema, senza le quali sarebbero compromesse le funzionalità di base dell'applicazione. Sono strettamente correlate a tutte le attività che un utente non registrato deve poter svolgere e costituiscono la base essenziale per il corretto funzionamento.

Media Priorità: Le specifiche assegnate a questa categoria sono importanti per il funzionamento dell'applicazione, ma possono essere implementate in un secondo momento. Sono principalmente legate alle attività che un utente registrato deve poter svolgere e aggiungono valore significativo all'esperienza complessiva dell'utente.

Bassa Priorità: Queste specifiche corrispondono a funzionalità marginali che, sebbene non influenzino il funzionamento complessivo della piattaforma, possono fornire vantaggi aggiuntivi o miglioramenti estetici. Saranno affrontate successivamente, una volta soddisfatte le esigenze di alta e media priorità.

Questa strategia di prioritizzazione consente di concentrare l'attenzione sulle componenti più critiche e indispensabili del progetto, garantendo un progresso graduale e mirato nell'implementazione delle funzionalità.

PRIORITA'	FUNZIONALITA'	CODICE
Alta Priorità	Visualizzazione Home page	UC1
	Bacheca e ricerca prodotti	UC2
	Registrazione Utente	UC3
	Login Utente	UC4
	Logout Utente	UC5
Media Priorità	Acquisto di un prodotto	UC6
	Ranking Utenti	UC7
	Creazione Carrello	
	Visualizzazione ordini effettuati	
	Visualizzazione prodotti messi in vendita	
	Visualizzazione prodotti venduti	
	Aggiunta recensione all'acquisto (con punteggio)	
	Ranking venditore	
	Classifica venditori	
	Eliminazione di un prodotto messo in vendita	
	Reso	
Bassa Priorità	Modifica delle specifiche di un prodotto messo in vendita	
	Sistema di Notifiche Personalizzate	
	Metodi di Pagamento Alternativi	

	Programma Fedeltà	
	Supporto Multilingue	
	Lista Desideri	
	Sistema di messaggistica Interna	

1.4 Analisi dell'architettura

Microservices based architecture

L'architettura basata su microservizi è un modello sofisticato che consente la costruzione di sistemi come una collezione di servizi, ognuno con una responsabilità ben definita.

Questo approccio si distingue per la suddivisione delle funzionalità in unità discrete e distribuibili in modo indipendente, permettendo a ciascuna di esse di poter essere sviluppata autonomamente.

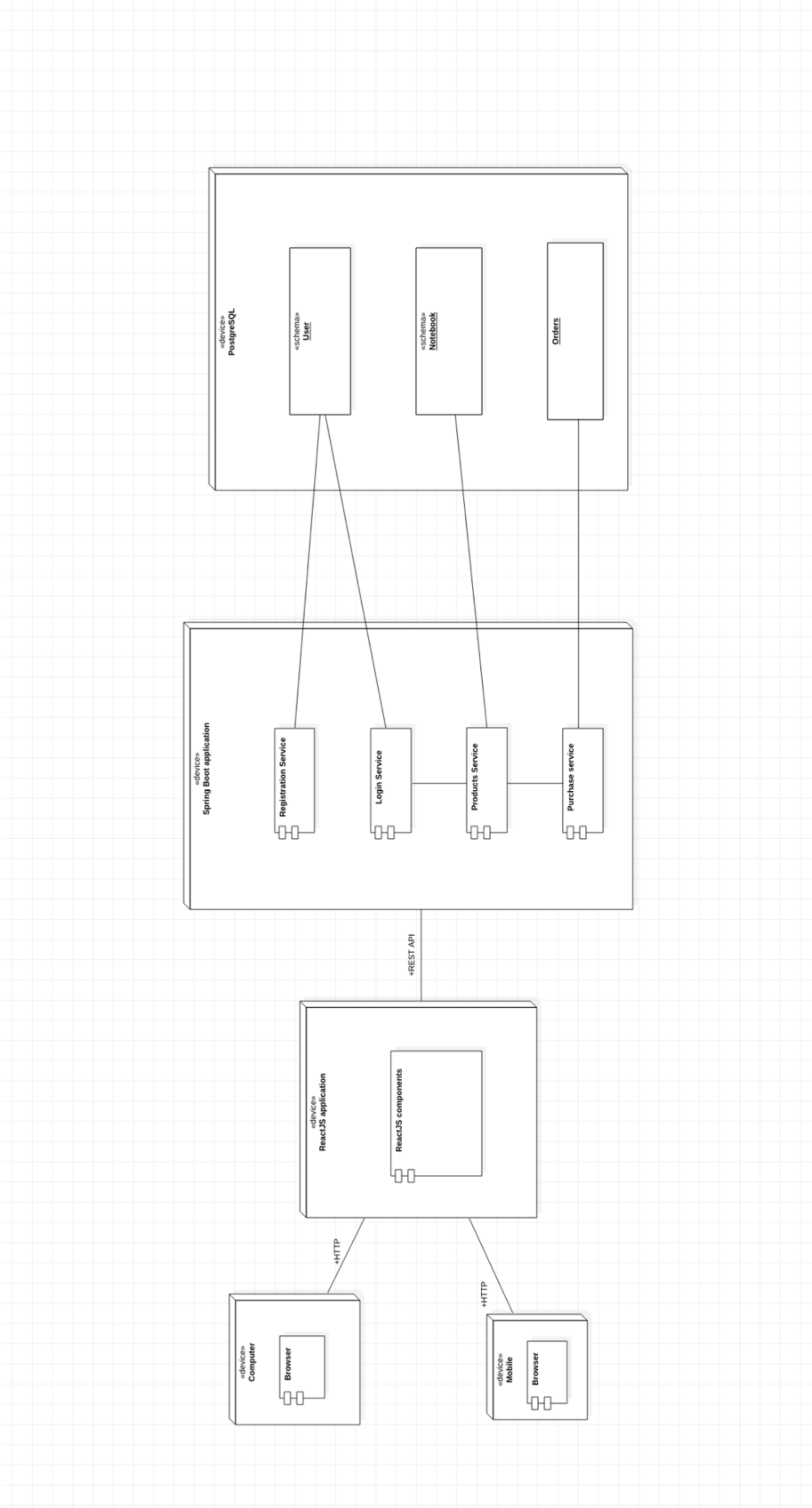
I microservizi, essendo autonomi, definiscono le proprie API consentendo l'esposizione delle loro funzionalità a componenti esterni. La comunicazione attraverso API è facilitata dall'uso di linguaggi comuni, come ad esempio REST.

I vantaggi principali dell'architettura a microservizi includono:

1. **Ridimensionamento:** La capacità di scalare singoli servizi in risposta a specifiche esigenze di carico di lavoro.
2. **Lavoro Simultaneo tra Gruppi di Sviluppo:** Diversi team possono lavorare contemporaneamente su microservizi indipendenti, accelerando lo sviluppo complessivo del sistema.
3. **Isolamento dei Guasti e Alta Manutenibilità:** La modularità consente il confinamento dei guasti a servizi specifici, agevolando la manutenzione e il debug.
4. **Scalabilità:** La possibilità di scalare solo i servizi che richiedono una maggiore capacità, anziché tutto il sistema.
5. **Indipendenza dallo Stack Tecnologico:** Ogni microservizio può essere sviluppato utilizzando diverse tecnologie, garantendo flessibilità nella scelta degli strumenti.
6. **Sicurezza:** La separazione dei servizi migliora la sicurezza complessiva del sistema.

In questo caso il sistema è stato progettato mappando ogni funzionalità con un microservizio sviluppato in Spring Boot. Questi microservizi sono accessibili tramite API REST, direttamente invocate dall'applicazione React, consentendo una comunicazione efficiente e modulare all'interno dell'architettura.

1.5 Deployment Diagram



2. Iterazione 1

2.1 Introduzione

Nella prima iterazione sono stati implementati i seguenti casi d'uso:

- **UC1: Visualizzazione Homepage**
- **UC2: Visualizzazione Bacheca**
 - UC2.1 Visualizzazione informazioni generali Notebook
 - UC2.2 Ricerca Notebook per campi
 - UC2.3 Inserimento Notebook nel carrello
- **UC3: Registrazione Utente**
- **UC4: Login Utente**
- **UC5 Logout Utente**

In seguito, viene riportata la descrizione dettagliata dei casi d'uso selezionati per la prima iterazione.

2.2 UC1 – Visualizzazione Homepage

Descrizione: All'apertura del sito web l'utente visualizza la pagina di benvenuto, qui potrà iniziare visualizzando delle informazioni utili per la navigazione nel sito, come:

- Link per la visualizzazione della bacheca, per la registrazioni e per il login.
- Iscrizione alla newsletter.
- Link per pagine di suggerimenti sul corretto utilizzo della piattaforma.

Attori Coinvolti: Utenti

Precondizione: Nessuna

Postcondizione: Visualizzazione corretta della Homepage

Svolgimento standard:

- a) L'utente si collega al sito Web.
- b) Una volta collegato, l'utente può esplorare la homepage.

2.3 UC2 – Visualizzazione Bacheca

Descrizione: La bacheca è la pagina dove gli utenti potranno visualizzare i prodotti disponibili all'acquisto inseriti da altri utenti, i prodotti saranno corredati dalle specifiche generali e potranno essere filtrati e ordinati in base a delle ricerche effettuate tramite i campi, inoltre se l'utente è loggato potrà inserirli nel proprio carrello.

Attori Coinvolti: Utenti

Precondizione: Accedere alla bacheca.

Postcondizione: Corretta visualizzazione e funzionamento della bacheca.

Svolgimento standard:

- a) L'utente si collega al sito Web e visualizza Homepage.
- b) L'utente clicca il link per essere reindirizzato alla bacheca.
- c) Il sistema mostra correttamente la bacheca:
 - a. Visualizzazione prodotti.
 - b. Ricerca e ordinamento tramite campi.
 - c. Bottone aggiungi al carrello.

UC2.1 - Visualizzazione informazioni generali Notebook

Descrizione: L'utente visualizza le specifiche tecniche dei prodotti in vendita ed il prezzo.

Attori Coinvolti: Utenti

Precondizione: Accedere alla bacheca.

Postcondizione: La bacheca mostra correttamente i prodotti in vendita.

Svolgimento standard:

- a) L'utente accede alla bacheca.
- b) L'utente visualizza i prodotti disponibili in vendita.

UC2.2 – Ricerca Notebook per campi

Descrizione: L'utente può effettuare una ricerca sui prodotti in bacheca utilizzando le specifiche tecniche dei prodotti come filtri.

Attori Coinvolti: Utenti

Precondizione: Accedere alla bacheca.

Postcondizione: La bacheca mostra correttamente i prodotti che soddisfano i criteri di ricerca specificati dall'utente.

Svolgimento standard:

- a) L'utente accede alla bacheca.
- b) L'utente visualizza i prodotti disponibili in vendita.
- c) L'utente inserisce nella form dedicata le specifiche desiderate ed effettua la ricerca.
- d) La bacheca mostra i prodotti filtrati secondo i parametri.

UC2.3 – Inserimento Notebook nel carrello

Descrizione: L'utente che si è registrato ed ha effettuato correttamente il Login potrà inserire nel proprio carrello un prodotto visualizzato nella bacheca cliccando l'apposito bottone. Nel caso l'utente non fosse loggato il sistema lo reindirizzerà alla pagina di Login.

Attori Coinvolti: Utenti

Precondizione: Utente Loggato.

Postcondizione: Il sistema inserisce il prodotto selezionato nel carrello personale dell'utente.

Trigger: L'utente clicca il pulsante "Aggiungi al carrello".

Svolgimento standard:

- a) L'utente loggato accede alla bacheca.

- b) L'utente loggato visualizza i prodotti disponibili in vendita.
- c) L'utente loggato clicca il pulsante "Aggiungi al carrello" del prodotto che vuole acquistare.

Svolgimento alternativo:

- a) L'utente non loggato accede alla bacheca.
- b) L'utente visualizza i prodotti disponibili in vendita.
- c) L'utente clicca il pulsante "Aggiungi al carrello" del prodotto che vuole acquistare.
- d) L'utente viene reindirizzato alla pagina di Login.

2.4 UC3 – Registrazione Utente

Descrizione: In fase di registrazione l'utente dovrà inserire obbligatoriamente i suoi dati personali (nome, cognome, mail, password) nell'apposito form di registrazione. Al termine della registrazione i suoi dati verranno salvati sul database e gli verrà assegnato un numero identificativo *ID*.

Attori Coinvolti: Utenti

Precondizione: Utente non registrato

Postcondizione: Utente registrato

Trigger: L'utente clicca il link "Register" sulla homepage.

Svolgimento standard:

- a) L'utente dalla homepage clicca sul link "Register" e viene reindirizzato alla pagina di registrazione.
- b) L'utente inserisce i propri dati nel form.
- c) L'utente clicca "Conferma".
- d) L'utente si è registrato correttamente.

Svolgimento alternativo:

- a) L'utente dalla homepage clicca sul link "Register" e viene reindirizzato alla pagina di registrazione.
- b) L'utente inserisce i propri dati nel form.
- c) Il sistema notifica all'utente che la mail inserita è già presente nel sistema.

2.5 UC4 – Login Utente

Descrizione: L'utente compila il form per il login; in caso di credenziali corrette il sistema consente l'accesso al servizio, altrimenti invita l'utente a riprovare.

Attori Coinvolti: Utenti

Precondizione: Utente non loggato.

Postcondizione: Utente Loggato.

Trigger: L'utente clicca il link "Login" sulla homepage.

Svolgimento standard:

- a) L'utente dalla homepage clicca sul link "Login" e viene reindirizzato alla pagina di Login.
- b) L'utente inserisce i propri dati nel form.
- c) L'utente clicca "Conferma".
- d) Il sistema controlla le credenziali inserite:
 - a. Se sono corrette, l'utente avviene il login e l'utente viene reindirizzato alla sua pagina personale.
 - b. Se sono errate, il sistema lo notifica all'utente e lo invita a riprovare.

2.6. UC5 – Logout Utente

Descrizione: L'utente può decidere di disconnettere volontariamente il suo account, oppure il logout verrà effettuato automaticamente alla chiusura dell'applicazione.

Attori Coinvolti: Utenti

Precondizione: Utente loggato.

Postcondizione: Utente non loggato.

Trigger: L'utente clicca sul tasto di logout oppure l'applicazione viene chiusa.

Svolgimento standard:

- a) L'utente clicca sull'icona per la gestione del proprio account e seleziona logout.
- b) Il sistema disconnette l'utente.

Svolgimento alternativo:

- a) L'utente chiude l'applicazione.
- b) Il sistema disconnette l'utente.

2.7 - Component Diagram

Utilizzando i casi d'uso identificati in questa iterazione come punto di partenza e applicando le euristiche di design, è stato possibile delineare l'architettura software seguendo il modello a microservizi.

In particolare, la progettazione si è basata su un'analisi approfondita dei casi d'uso selezionati, cercando di adottare le migliori pratiche euristiche di design.

L'obiettivo è stato sviluppare un'architettura che riflettesse in modo efficace le funzionalità richieste, garantendo al contempo una struttura modulare e scalabile grazie al paradigma a microservizi.

Questo approccio non solo favorisce la chiarezza nell'organizzazione delle funzionalità del sistema, ma anche la facilità di manutenzione e futuri sviluppi.

La scelta di modellare l'architettura come un insieme di microservizi mira a ottenere una maggiore flessibilità e adattabilità, elementi cruciali soprattutto in contesti in cui le esigenze possono evolvere nel tempo.

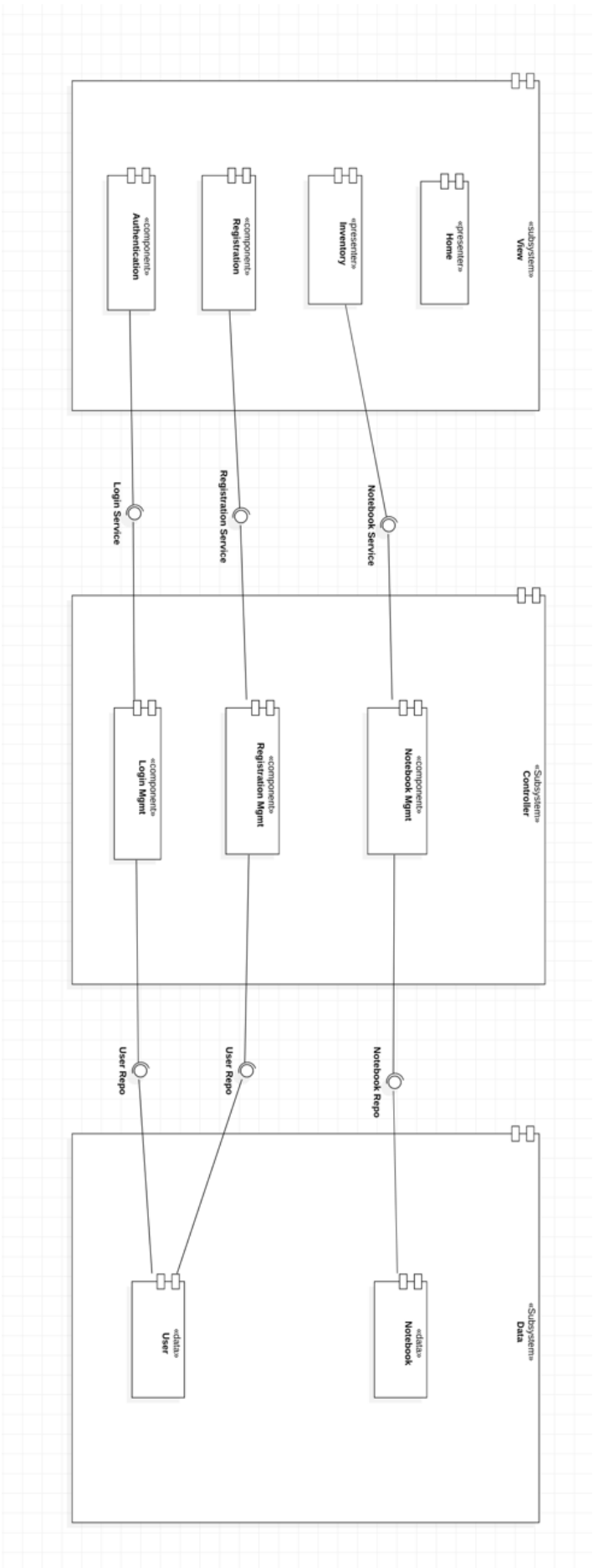
Un UML Component Diagram è un tipo di diagramma strutturale che rappresenta l'architettura e la struttura di un sistema software in termini di componenti e delle relazioni tra di essi.

I componenti in questo contesto possono essere visti come moduli o unità software indipendenti, che incapsulano funzionalità specifiche.

Il Component Diagram mostra come le varie parti di un sistema software interagiscono e collaborano per raggiungere gli obiettivi dell'applicazione.

I componenti sono rappresentati graficamente come blocchi. Le relazioni tra i componenti, come le dipendenze o le associazioni, sono visualizzate attraverso linee di connessione.

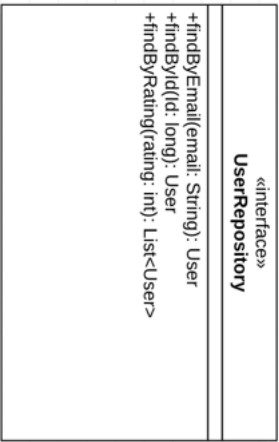
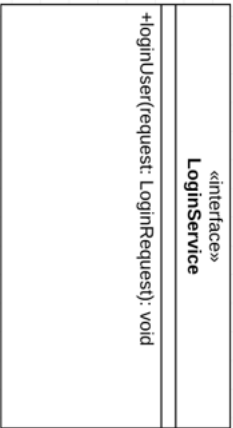
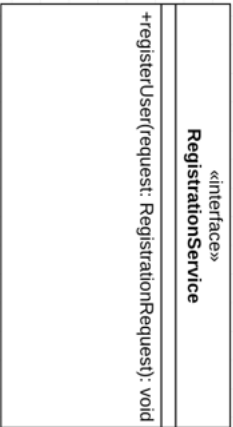
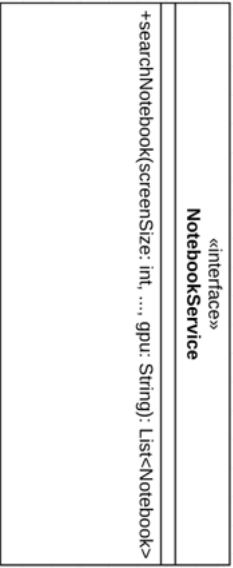
Questo tipo di diagramma è particolarmente utile per comprendere la struttura del sistema e le dipendenze tra i vari moduli.



2.8 - UML Class Diagram per le interfacce

Il diagramma UML delle classi per le interfacce offre una rappresentazione visiva delle diverse interfacce presenti nel sistema, evidenziando le rispettive funzioni e le relazioni con i dati di input e output.

La figura fornisce una panoramica chiara e dettagliata delle interfacce utilizzate nel contesto del sistema, grazie a questo diagramma risulterà più facile comprendere come le varie componenti interagiscono tra loro e quali dati vengono scambiati.



2.9 - UML Class Diagram per i tipi di dato

Il diagramma UML delle classi per i tipi di dato offre una rappresentazione visiva chiara e dettagliata dei diversi tipi di dato necessari per lo sviluppo dell'applicazione.

La figura illustra i vari tipi di dato utilizzati nel contesto dell'applicazione, facilitando la comprensione della struttura e delle relazioni tra di essi.

Questo diagramma facilita la comprensione e la documentazione dei tipi di dato essenziali per lo sviluppo dell'applicazione, contribuendo così a una progettazione chiara e organizzata del sistema.

2.10 – Testing

2.10.1 – Analisi Statica

L'analisi statica rappresenta una fase cruciale nel processo di sviluppo del software, in quanto consente di esaminare il codice sorgente senza eseguirlo, identificando potenziali errori, inconsistenze o violazioni delle best practices.

Per effettuare l'analisi statica è stata utilizzata l'estensione Language Support for Java(TM) by Red Hat fornita dall'IDE Visual Studio Code.

L'utilizzo di un'estensione specifica, come quella fornita da Red Hat per Java, fornisce strumenti avanzati per condurre un'analisi approfondita del codice, migliorando la qualità complessiva del software.

2.10.2 – API Test

La fase di testing delle API è un passaggio cruciale nello sviluppo del software, in cui si verifica la corretta implementazione e il comportamento delle interfacce di programmazione delle applicazioni.

Per questa specifica attività, è stato adottato il software Insomnia.

Insomnia è una piattaforma di test API che consente agli sviluppatori di eseguire in modo efficiente test su richieste API, verificare risposte e analizzare i risultati.

Insomnia offre un'interfaccia utente intuitiva che semplifica la configurazione e l'esecuzione di test API. Attraverso questa piattaforma, gli sviluppatori possono creare richieste personalizzate, gestire ambienti di test e ottenere una chiara visualizzazione delle risposte API.

L'utilizzo di Insomnia per la fase di testing delle API garantisce una valutazione accurata delle funzionalità delle interfacce, facilitando il rilevamento di eventuali problemi o inefficienze prima della distribuzione del software.

Nell'iterazione 1 sono state testate le API sviluppate per gestire i Notebook e gli Utenti, in particolare:

- Inserimento di un Notebook nell'applicazione.
- Eliminazione di un Notebook
- Ricerca di un Notebook in base ai suoi campi.
- Inserimento di un utente nell'applicazione.

The screenshot displays the Kong API Gateway interface. On the left, a sidebar lists various API endpoints under the 'Progetto Pac' environment. The main panel shows a GET request to `localhost:8080/notebook/byId?id=4` with a 'Send' button. Below the URL, there are sections for 'QUERY PARAMETERS', 'PATH PARAMETERS', and 'BODY'. The 'Preview' tab on the right shows the response status '200 OK' with a response time of '8.77 ms' and a body size of '198 B'. The response body is a JSON object representing a notebook.

```
1 {
2   "id": 4,
3   "nameModel": "HP Envy",
4   "brand": "HP",
5   "operatingSystem": "WINDOWS",
6   "screenSize": 15,
7   "ram": 8,
8   "disk": "SSD",
9   "storage": 1000,
10  "cpu": "Intel i5",
11  "gpu": "NVIDIA",
12  "description": "HP Envy 2022",
13  "price": 1500.0
14 }
```

At the bottom of the interface, there is a status bar indicating 'Online' and 'Made with ❤ by Kong'.

- API per la ricerca di un notebook tramite storage

GET localhost:8080/notebook/byStorage?storage=512 Send 200 OK 5.45 ms 215 B 1 Year Ago

Parameters Body Auth Headers Docs

URL PREVIEW
http://localhost:8080/notebook/byStorage?storage=512

QUERY PARAMETERS Import from URL Bulk Edit

Add Delete All Toggle Description

Name Value

PATH PARAMETERS

Path parameters are url path segments that start with a colon ':' e.g. ':id'

Preview Headers Cookies Timeline

```
1 [
2 {
3   "id": 1,
4   "nameModel": "AppleSiliconM1",
5   "brand": "APPLE",
6   "operatingSystem": "MACOS",
7   "screenSize": 13,
8   "ram": 8,
9   "disk": "SSD",
10  "storage": 512,
11  "cpu": "Chip M1",
12  "gpu": "APPLE",
13  "description": "Apple Silicon m1 2021",
14  "price": 1200.0
15 }
16 ]
```

- API per la ricerca di un notebook tramite prezzo massimo

GET localhost:8080/notebook/byMaxPrice?price=1000 Send 200 OK 7.51 ms 217 B 1 Year Ago

Parameters Body Auth Headers Docs

URL PREVIEW
http://localhost:8080/notebook/byMaxPrice?price=1000

QUERY PARAMETERS Import from URL Bulk Edit

Add Delete All Toggle Description

Name Value

PATH PARAMETERS

Path parameters are url path segments that start with a colon ':' e.g. ':id'

Preview Headers Cookies Timeline

```
1 [
2 {
3   "id": 3,
4   "nameModel": "Asus F564gths",
5   "brand": "ASUS",
6   "operatingSystem": "WINDOWS",
7   "screenSize": 17,
8   "ram": 8,
9   "disk": "HDD",
10  "storage": 1000,
11  "cpu": "Intel I7",
12  "gpu": "NVIDIA",
13  "description": "Asus usato pochissimo",
14  "price": 900.0
15 }
16 ]
```

- API per l'inserimento di un Notebook

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** localhost:8080/notebook/insert
- Status:** 200 OK
- Response Time:** 48.6 ms
- Response Size:** 31 B
- Timestamp:** 11 Months Ago
- Request Body (JSON):**

```
1 {
2   "nameModel": "Hp Envy",
3   "brand": "HP",
4   "operatingSystem": "WINDOWS",
5   "screenSize": 15,
6   "ram": 16,
7   "disk": "SSD",
8   "storage": 512,
9   "cpu": "Intel i7",
10  "gpu": "NVIDIA GEFORCE",
11  "description": "Hp Envy Nuovo ancora in confezione",
12  "price": 1600.0
13 }
```
- Response Body (Text):**

```
1 Notebook correttamente inserito
```


2.10.1 – JUnit Test

JUnit è un framework di testing per Java che facilita la scrittura e l'esecuzione di test unitari automatizzati.

I test JUnit sono implementati attraverso annotazioni e metodi di verifica, contribuendo a garantire la correttezza e l'affidabilità del codice Java.

In questa iterazione è stata testata la funzione *getByEmail()* della classe *UserService*, questa funzione serve per ricercare un utente salvato nel database tramite mail e viene sfruttata abbondantemente all'interno dell'applicazione.

```
@Test
public void getByEmailTest(){
    //Creo un utente e lo salvo
    String name = "Francesco";
    String lastName = "Rossi";
    String email = "francesco.rossi@gmail.com";
    String password = "passWord";

    User expected = new User(name, lastName, email, password);
    userRepository.save(expected);

    //Effettuo la ricerca
    Optional<User> optionalResult = userService.getByEmail(email);
    //Controlla che ci sia un utente
    assertTrue(optionalResult.isPresent());
    User actual = optionalResult.get();
    //Controllo che sia l'utente giusto
    assertEquals(expected.getUserId(), actual.getUserId());
}
```

Codice 1

In questa iterazione è stata anche testata la funzione *findByOperatingSystem()* della classe NotebookService, questa funzione viene utilizzata dall'applicazione per ricercare e filtrare i notebook in base al sistema operativo, è una funzionalità essenziale dell'applicazione in quanto viene sfruttata per offrire i servizi della bacheca e ricerca prodotti.

```
@Test
public void findByOperatingSystemTest(){

    OperatingSystem windows = OperatingSystem.WINDOWS;
    OperatingSystem macos = OperatingSystem.MACOS;

    //Ritorna una lista di notebook con sistema operativo windows
    List<Notebook> listWindows= notebookRepository.findByOperatingSystem(windows);

    //Controllo che per ogni elemento restituito il sistema operativo sia windows
    for(Notebook notebook : listWindows){
        assertEquals(windows, notebook.getOperatingSystem());
    }

    //Ritorna una lista di notebook con sistema operativo windows
    List<Notebook> listMacos =notebookRepository.findByOperatingSystem(macos);

    //Controllo che per ogni elemento restituito il sistema operativo sia macos
    for(Notebook notebook : listMacos){
        assertEquals(macos, notebook.getOperatingSystem());
    }
}
```

Codice 2

3. Iterazione 2

3.1 Introduzione

Nella seconda iterazione sono stati implementati i seguenti casi d'uso:

- **UC6: Acquisto di un prodotto**
- **UC7: Sviluppo e visualizzazione Ranking**

In seguito, viene riportata la descrizione dettagliata dei casi d'uso selezionati per la seconda iterazione.

3.2 UC6 – Acquisto di un prodotto

Descrizione: All'interno della bacheca dell'applicazione, l'utente ha la possibilità di acquistare il prodotto desiderato cliccando sul pulsante "Acquista".

L'utente verrà reindirizzato nella pagina di completamento ordine, dove per finalizzare l'acquisto dovrà indicare il metodo di pagamento, l'indirizzo di consegna e se desidera altre indicazioni utili.

Una volta avvenuta la transazione il notebook acquistato viene rimosso dalla bacheca e l'ordine salvato nel database.

Trigger: Utente preme il pulsante "Acquista" su un prodotto.

Attori Coinvolti: Utenti.

Precondizione: Utente loggato.

Postcondizione: Aggiunta dell'ordine al database e rimozione del prodotto dalla bacheca.

Svolgimento standard:

- a) L'utente naviga all'interno della bacheca e preme il tasto "Acquista" su un prodotto.
- b) L'utente viene reindirizzato nella pagina di completamento ordine.
- c) L'utente fornisce il metodo di pagamento, l'indirizzo di consegna ed eventuali note.

- d) Una volta effettuata la transazione l'ordine viene inserito nel database ed il prodotto rimosso dalla bacheca.

Svolgimento alternativo:

- a) L'utente naviga all'interno della bacheca e preme il tasto "Acquista" su un prodotto.
- b) L'utente viene reindirizzato nella pagina di completamento ordine.
- c) L'utente fornisce il metodo di pagamento, l'indirizzo di consegna ed eventuali note.
- d) L'utente fornisce dati non validi o la transazione viene rifiutata.
- e) L'utente viene reindirizzato in bacheca.

3.3.UC7 - Sviluppo e visualizzazione algoritmo di Ranking

Descrizione: Per permettere la visualizzazione dei venditori più affidabili, l'applicazione fornisce una pagina dove è possibile consultare un ranking sviluppato tramite un algoritmo che classifica gli utenti in base alle recensioni ed al numero di vendite effettuate.

Attori Coinvolti: Utenti.

Trigger: Richiesta visualizzazione pagina ranking.

Precondizione: Nessuna

Postcondizione: Visualizzazione della classifica utenti.

Svolgimento standard:

- a) L'utente clicca visualizzazione pagina Ranking.

3.4 - Algoritmo di Ranking

Breve descrizione: L'algoritmo di ranking è stato sviluppato con l'obiettivo di creare una classifica degli utenti basata sull'affidabilità dei venditori all'interno del sistema.

Ogni volta che un acquirente completa un acquisto, fornisce una valutazione dell'ordine, elemento cruciale nel calcolo del rating di un utente venditore.

L'algoritmo utilizza un approccio di tipo Merge Sort per ordinare gli utenti in base ai loro rating, assicurando così un elenco classificato che riflette la qualità e l'affidabilità delle transazioni effettuate.

Questo metodo di ordinamento contribuisce a promuovere una maggiore trasparenza e fiducia all'interno della piattaforma, mettendo in evidenza i venditori più affidabili e riconosciuti dagli acquirenti.

Attori coinvolti: Utenti e Sistema.

Trigger: Al completamento di un ordine.

Postcondizione: Classifica aggiornata degli utenti.

Passi dell'algoritmo:

1. Ogni volta che viene completato un ordine, il rating dell'utente venditore viene aggiornato.
2. L'algoritmo viene eseguito per aggiornare la classifica:

1. Verifica della Condizione di Fine Ricorsione:

Se l'intervallo di indice `left` è maggiore o uguale all'indice `right`, l'intervallo contiene zero o un elemento e non è necessario ordinare. In questo caso, ritorna senza fare nulla.

2. Calcolo del Punto Medio:

Calcola l'indice del punto medio come $(left + right) / 2$. Questo dividerà l'intervallo corrente in due parti più piccole.

3. Chiamata Ricorsiva per la Parte Sinistra:

Chiama ricorsivamente ``mergeSort`` per la parte sinistra dell'intervallo, cioè da ``left`` a ``middle``.

4. Chiamata Ricorsiva per la Parte Destra:

Chiama ricorsivamente ``mergeSort`` per la parte destra dell'intervallo, cioè da ``middle + 1`` a ``right``.

5. Fase di Merge:

Dopo che le due sottoparti sono ordinate separatamente, esegue la fase di merge.

Crea due array temporanei (``leftArray`` e ``rightArray``) per memorizzare le sottoparti ordinate.

Inizia a confrontare gli elementi di ``leftArray`` e ``rightArray``, prendendo l'elemento più piccolo e posizionandolo nell'array principale (``userList``).

Continua a farlo finché tutti gli elementi di entrambi gli array temporanei sono stati inseriti nell'array principale.

6. Copiatura degli Elementi Rimanenti:

Se ci sono elementi rimanenti in uno dei due array temporanei, copia gli elementi rimanenti nell'array principale.

7. Fine Ricorsione:

Ogni chiamata ricorsiva raggiunge il suo punto di terminazione quando l'intervallo considerato è diventato sufficientemente piccolo da non richiedere ulteriori suddivisioni. L'algoritmo di merge sort prosegue con questo processo di divisione e fusione ricorsiva fino a quando l'intera sequenza è ordinata.

L'efficienza di merge sort è sostenuta dal fatto che l'operazione di fusione può essere eseguita in modo efficiente su sequenze già ordinate, contribuendo così alla performance globale dell'algoritmo.

3.4.1 – Pseudocodice e analisi complessità algoritmo

```
MergeSort(userList, left, right)
    if left >= right
        return

    middle = (left + right) / 2

    // Chiamata ricorsiva per la parte sinistra
    MergeSort(userList, left, middle)

    // Chiamata ricorsiva per la parte destra
    MergeSort(userList, middle + 1, right)

    // Fase di merge
    Merge(userList, left, middle, right)

// Ogni chiamata ricorsiva divide l'array a metà
// Numero totale di chiamate ricorsive:  $\log_2(n)$ 
// Ogni chiamata ricorsiva richiede tempo  $O(n)$  per l'esecuzione della fase di // merge
// Complessità totale di MergeSort:  $O(n \log n)$ 

Merge(userList, left, middle, right)
    n1 = middle - left + 1
    n2 = right - middle

    // Creazione di array temporanei
    leftArray[n1], rightArray[n2]

    // Copia dei dati negli array temporanei
    for i = 0 to n1 - 1
        leftArray[i] = userList[left + i]
    for j = 0 to n2 - 1
        rightArray[j] = userList[middle + 1 + j]

    // Fase di merge
    i = 0
    j = 0
    k = left

    while i < n1 && j < n2
        if leftArray[i].getRating() >= rightArray[j].getRating()
            userList[k] = leftArray[i]
            i = i + 1
        else
            userList[k] = rightArray[j]
            j = j + 1
        k = k + 1
```



```
// Copia degli elementi rimanenti di leftArray
```

```
while i < n1
```

```
    userList[k] = leftArray[i]
```

```
    i = i + 1
```

```
    k = k + 1
```

```
// Copia degli elementi rimanenti di rightArray
```

```
while j < n2
```

```
    userList[k] = rightArray[j]
```

```
    j = j + 1
```

```
    k = k + 1
```

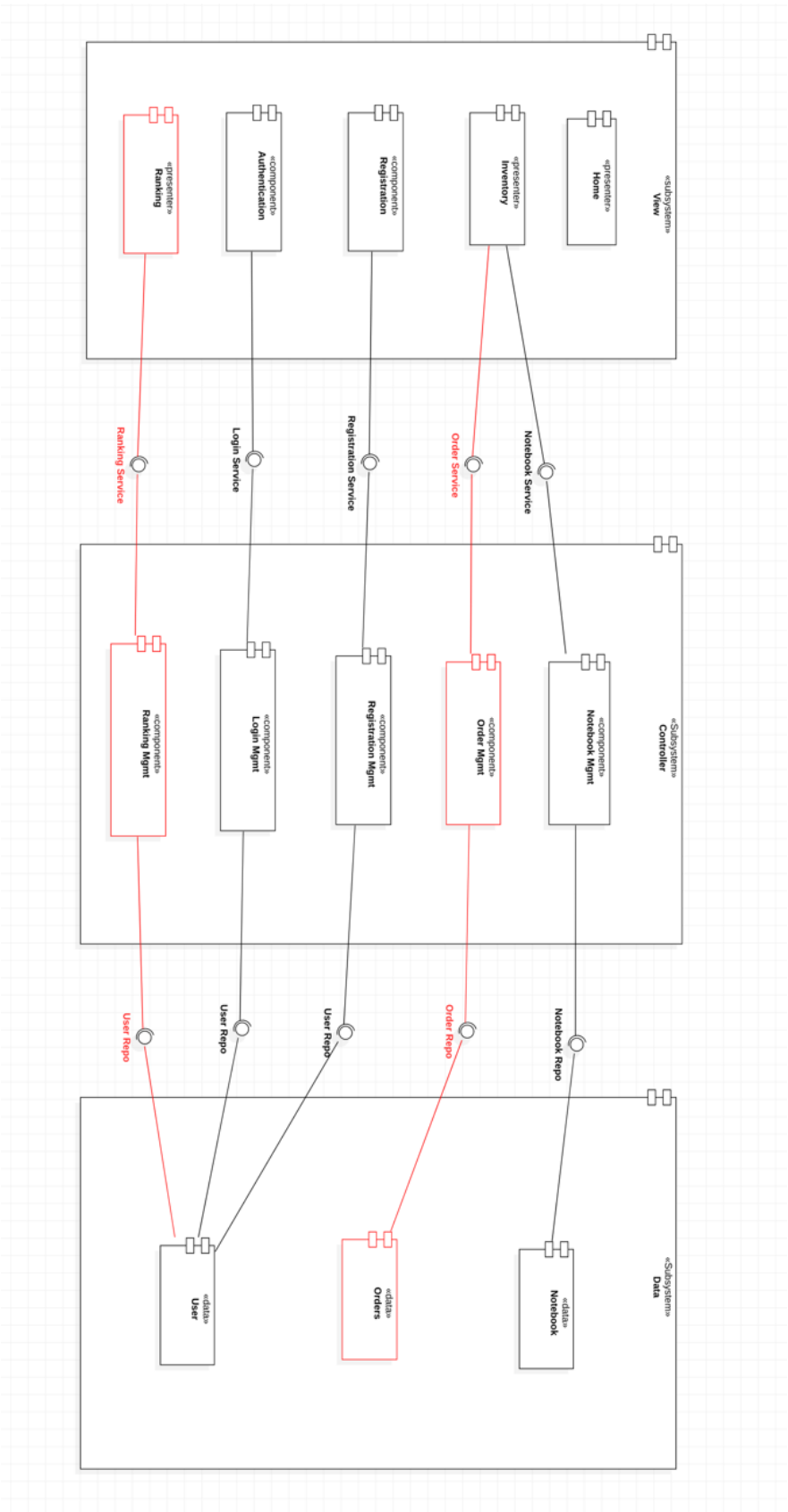
// La fase di merge richiede tempo lineare $O(n)$, dove "n" è la somma delle dimensioni degli array da unire

// Nella peggior situazione, "n" è la dimensione totale dell'array da ordinare

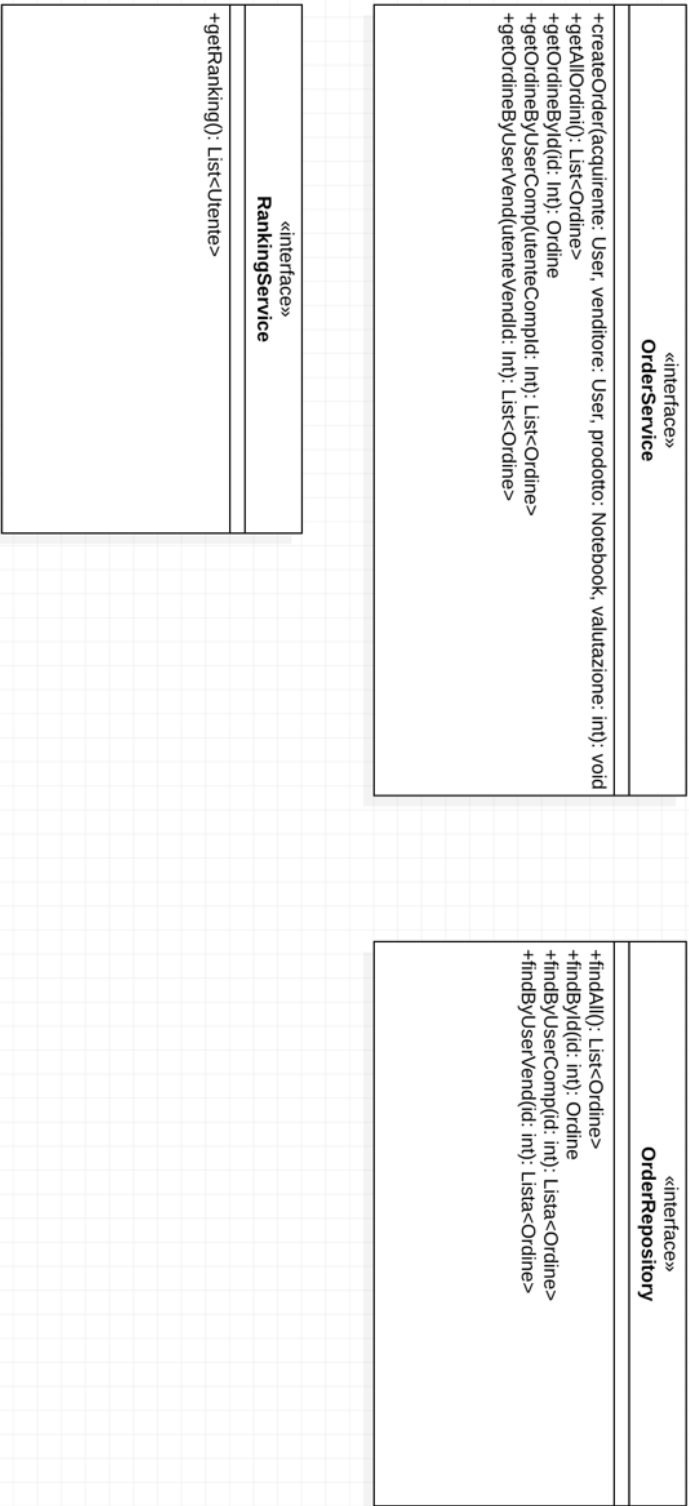
// Complessità di Merge: $O(n)$

Codice 3

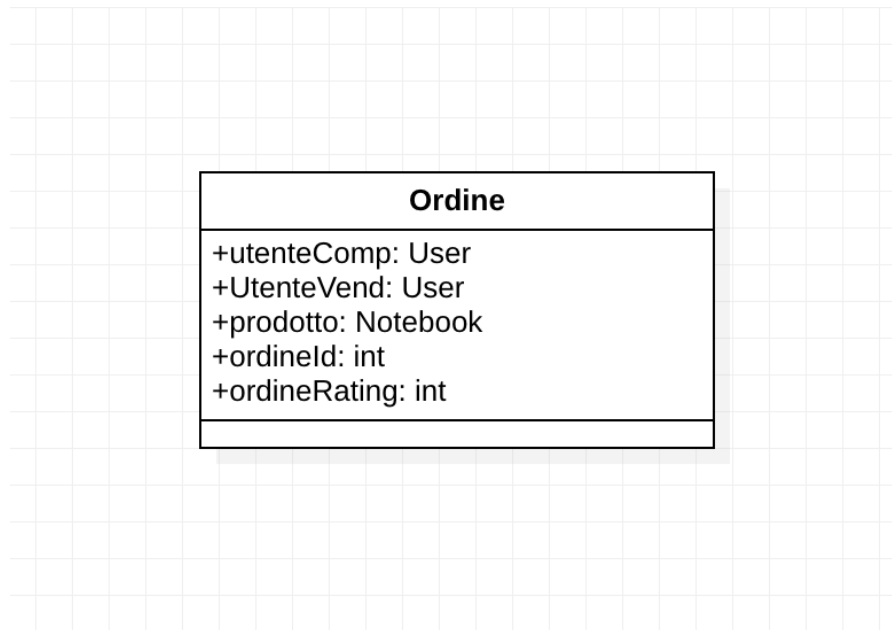
3.5 - UML Component Diagram



3.6 - UML Class Diagram per le interfacce



3.7 - UML Class Diagram per i tipi di dato



3.8 – Testing

3.8.1 – Analisi Statica

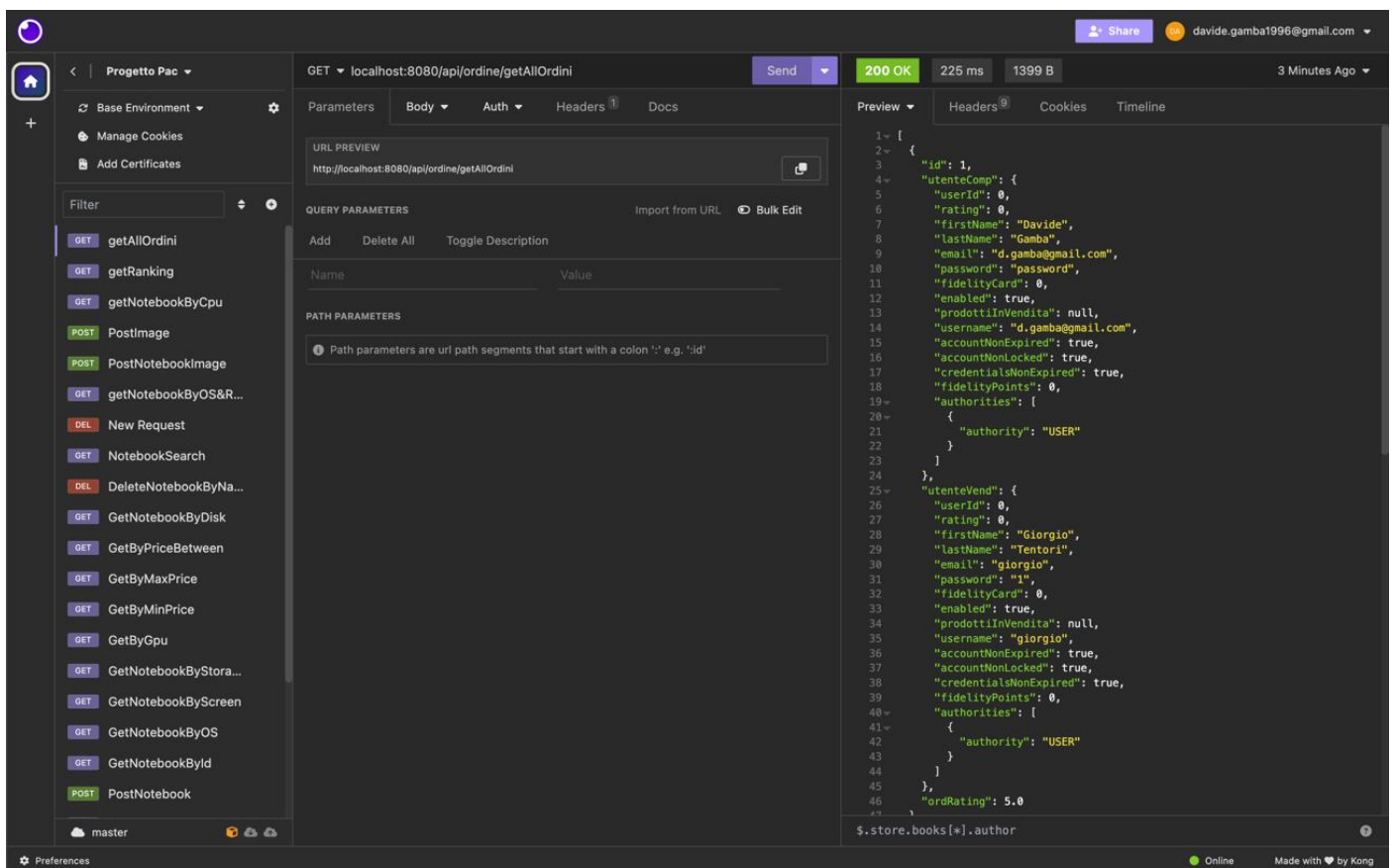
Per effettuare l'analisi statica è stata utilizzata l'estensione Language Support for Java(TM) by Red Hat fornita dall'IDE Visual Studio Code.

3.8.2 – API Test

Per la fase di testing delle API è stato utilizzato il software Insomnia.

Nell'iterazione 2 sono state testate le API sviluppate per gestire gli Ordini e la vista generata dall'algoritmo di Ranking, in particolare:

- Restituzione di tutti gli ordini.
- Restituzione classifica utenti.



- API per la restituzione degli ordini

The screenshot displays a REST client interface with the following components:

- Request Bar:** Method `GET`, URL `localhost:8080/api/ordine/getAllOrdini`, and a `Send` button.
- Status Bar:** `200 OK`, `225 ms`, `1399 B`, and `26 Minutes Ago`.
- Parameters Panel:** Includes tabs for `Parameters`, `Body`, `Auth`, `Headers`, and `Docs`.
 - URL PREVIEW:** Shows the request URL `http://localhost:8080/api/ordine/getAllOrdini`.
 - QUERY PARAMETERS:** Includes buttons for `Add`, `Delete All`, and `Toggle Description`.
 - PATH PARAMETERS:** Includes a note: "Path parameters are url path segments that start with a colon ':' e.g. ':id'".
- Response Panel:** Includes tabs for `Preview`, `Headers`, `Cookies`, and `Timeline`. The `Preview` tab shows the JSON response:

```
1 [
2 {
3   "id": 1,
4   "utenteComp": {
5     "userId": 0,
6     "rating": 0,
7     "firstName": "Davide",
8     "lastName": "Gamba",
9     "email": "d.gamba@gmail.com",
10    "password": "password",
11    "fidelityCard": 0,
12    "enabled": true,
13    "prodottiInVendita": null,
14    "username": "d.gamba@gmail.com",
15    "accountNonExpired": true,
16    "accountNonLocked": true,
17    "credentialsNonExpired": true,
18    "fidelityPoints": 0,
19    "authorities": [
20      {
21        "authority": "USER"
22      }
23    ]
24  },
25   "utenteVend": {
26     "userId": 0,
27     "rating": 0,
28     "firstName": "Giorgio",
29     "lastName": "Tentori",
30     "email": "giorgio",
31     "password": "1",
32     "fidelityCard": 0,
33     "enabled": true,
34     "prodottiInVendita": null,
35     "username": "giorgio",
36     "accountNonExpired": true,
```

- API per la restituzione Ranking

GET localhost:8080/ranking/getRanking Send 200 OK 40.8 ms 986 B 1 Day Ago

Parameters Body Auth Headers Docs

URL PREVIEW
http://localhost:8080/ranking/getRanking

QUERY PARAMETERS Import from URL Bulk Edit
Add Delete All Toggle Description

Name	Value
------	-------

PATH PARAMETERS
Path parameters are url path segments that start with a colon ':' e.g. ':id'

Preview Headers Cookies Timeline

```
1 [
2   {
3     "userId": 2,
4     "rating": 10,
5     "firstName": "Giorgio",
6     "lastName": "Tentori",
7     "email": "giorgio",
8     "password": "1",
9     "fidelityCard": 0,
10    "enabled": true,
11    "prodottiInVendita": [],
12    "username": "giorgio",
13    "accountNonExpired": true,
14    "accountNonLocked": true,
15    "credentialsNonExpired": true,
16    "fidelityPoints": 0,
17    "authorities": [
18      {
19        "authority": "USER"
20      }
21    ]
22  },
23  {
24    "userId": 3,
25    "rating": 9,
26    "firstName": "Aurora",
27    "lastName": "Zanenga",
28    "email": "a.zanenga@gmail.com",
29    "password": "password",
30    "fidelityCard": 0,
31    "enabled": true,
32    "prodottiInVendita": [],
33    "username": "a.zanenga@gmail.com",
34    "accountNonExpired": true,
35    "accountNonLocked": true,
36    "credentialsNonExpired": true,
37    "fidelityPoints": 0,
38    "authorities": [
39      {
40        "authority": "USER"
41      }
42    ]
43  },
44  {
45    "userId": 1,
46    "rating": 8,
47    "firstName": "Maurizio",
48    "lastName": "Maurizio",
49    "email": "maurizio",
50    "password": "maurizio",
51    "fidelityCard": 0,
52    "enabled": true,
53    "prodottiInVendita": [],
54    "username": "maurizio",
55    "accountNonExpired": true,
56    "accountNonLocked": true,
57    "credentialsNonExpired": true,
58    "fidelityPoints": 0,
59    "authorities": [
60      {
61        "authority": "USER"
62      }
63    ]
64  }
65 ]
```

3.8.3 – JUnit Test

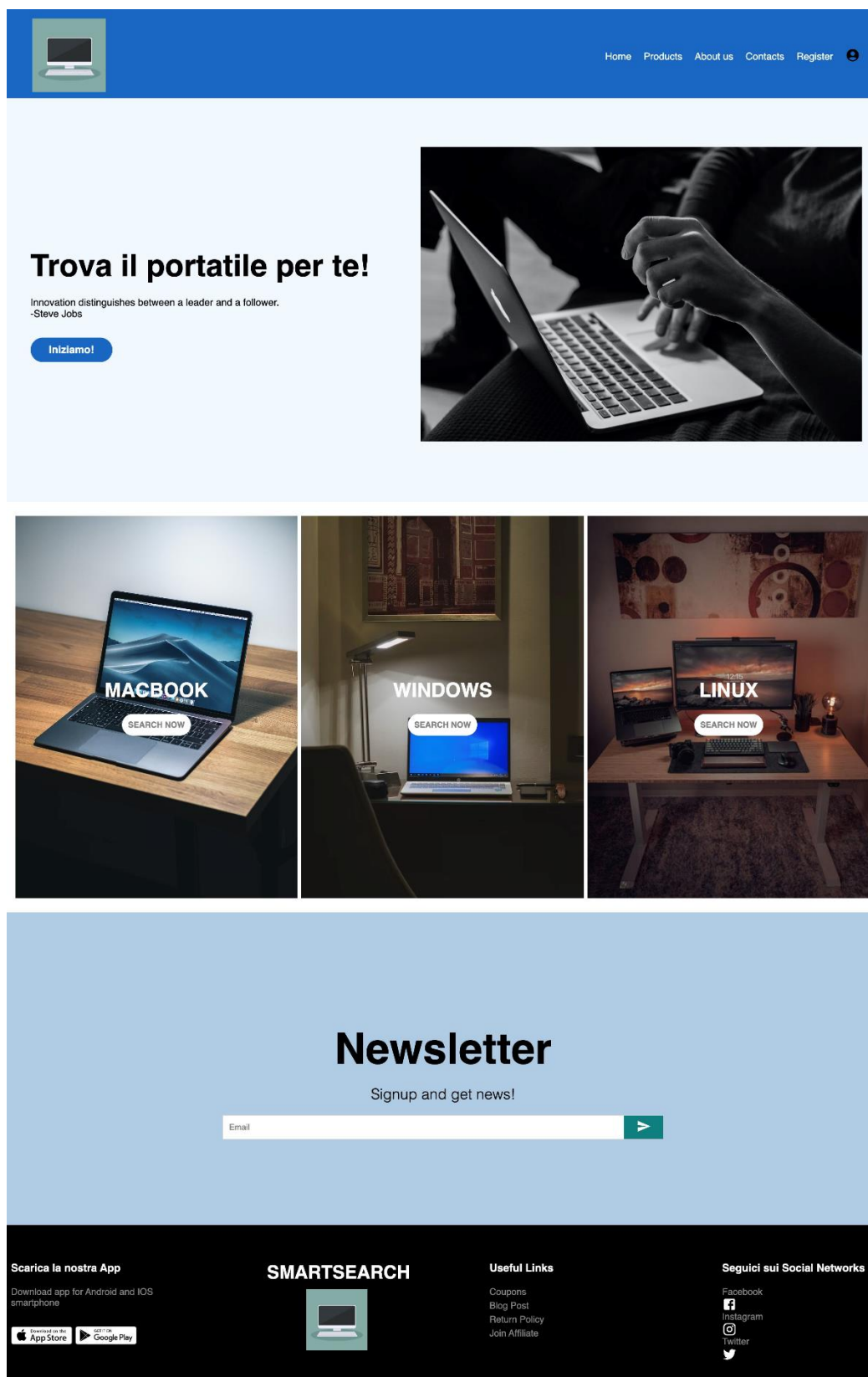
In questa iterazione è stata testata la funzione *getOrdine()* della classe *OrderService*, questa funzione serve per restituire un ordine salvato nel database tramite una ricerca basata sul campo id dell'ordine.

```
@Test
public void getOrdineTest(){
    int expectedId = 1;
    //Effettuo una ricerca per Id sugli ordini inseriti
    Optional<Ordine> actualOptOrdine = ordineService.getOrdine(expectedId);
    //Controllo che abbia restituito un ordine
    assertTrue(actualOptOrdine.isPresent());
    Ordine actualOrdine = actualOptOrdine.get();
    //Controllo che l'ordine corrisponda a quello cercato
    assertEquals(expectedId, actualOrdine.getId());
}
```


Codice 4

3.9 – Screenshot Applicazione

3.9.1 Homepage



3.9.2 Bacheca



HomeProductsAbout usContactsRegister

Black Friday! Fino al 70% sui prodotti in catalogo!

Brand

Operating System

Screen Size

RAM

Storage

Type of Disk

CPU

Insert CPU

GPU

Price

Min

Max


Ricerca prodotti:

PROMO OUTLET: Scopri tutte le categorie con prodotti scontati fino al 50%

Scopri i migliori notebook dalle dimensioni ridotte, ideali per chi desidera portare il proprio dispositivo sempre con sé.

Vuoi acquistare un pc portatile? Scopri tutte le caratteristiche principali di un notebook.

Macboook Pro M1



Processore
Chip M1

Dimensione memoria RAM
8

Capacità di memorizzazione totale in GB
512 GB

Sistema Operativo
MACOS

Marca
APPLE

Scheda Grafica
APPLE

Tipo di memoria
SSD


Grandezza dello schermo
13 pollici

Prezzo:
1200

Per modalità e tempi di consegna, contattare il venditore, SmartSearch non si assume alcune responsabilità sulle modalità di consegna del prodotto.

Aggiungi al carrello

Lenovo Thinkpad



Processore
Intel i5

Dimensione memoria RAM
16

Capacità di memorizzazione totale in GB
1000 GB

Sistema Operativo
WINDOWS

Marca
LENOVO

Scheda Grafica
NVIDIA

Tipo di memoria
SSD


Grandezza dello schermo
15 pollici

Prezzo:
1300

Per modalità e tempi di consegna, contattare il venditore, SmartSearch non si assume alcune responsabilità sulle modalità di consegna del prodotto.

Aggiungi al carrello

Asus F564GTHS



Processore
Intel i7

Dimensione memoria RAM
8

Capacità di memorizzazione totale in GB
1000 GB

Sistema Operativo
WINDOWS

Marca
ASUS

Scheda Grafica
NVIDIA

Tipo di memoria
HDD

Grandezza dello schermo
17 pollici

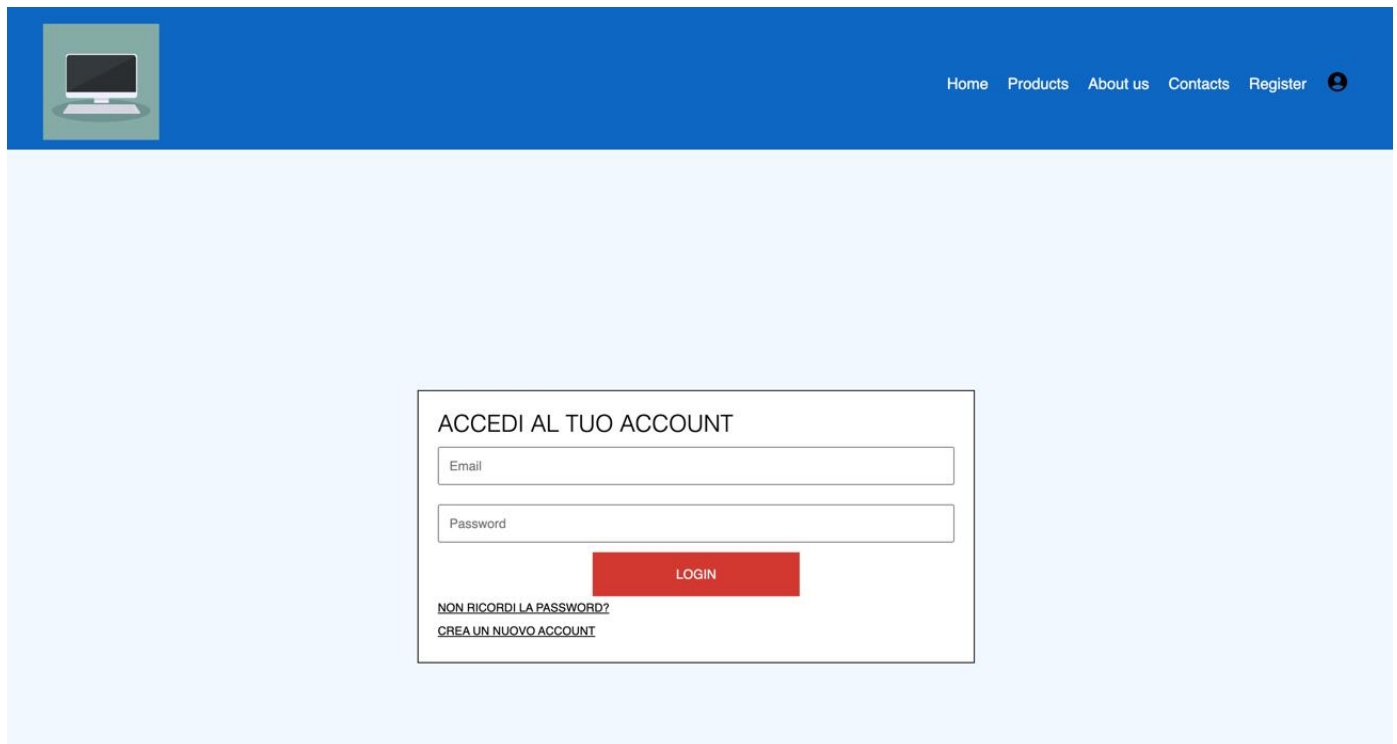
Prezzo:
900

Per modalità e tempi di consegna, contattare il venditore, SmartSearch non si assume alcune responsabilità sulle modalità di consegna del prodotto.

Aggiungi al carrello

47

3.9.3 Login Page



The image shows a login page with a blue header. On the left of the header is a laptop icon. On the right are links: Home, Products, About us, Contacts, Register, and a user profile icon. The main content area is light blue and contains a white login form. The form has the title 'ACCEDI AL TUO ACCOUNT', followed by input fields for 'Email' and 'Password', and a red 'LOGIN' button. Below the button are two links: 'NON RICORDI LA PASSWORD?' and 'CREA UN NUOVO ACCOUNT'.

ACCEDI AL TUO ACCOUNT

Email

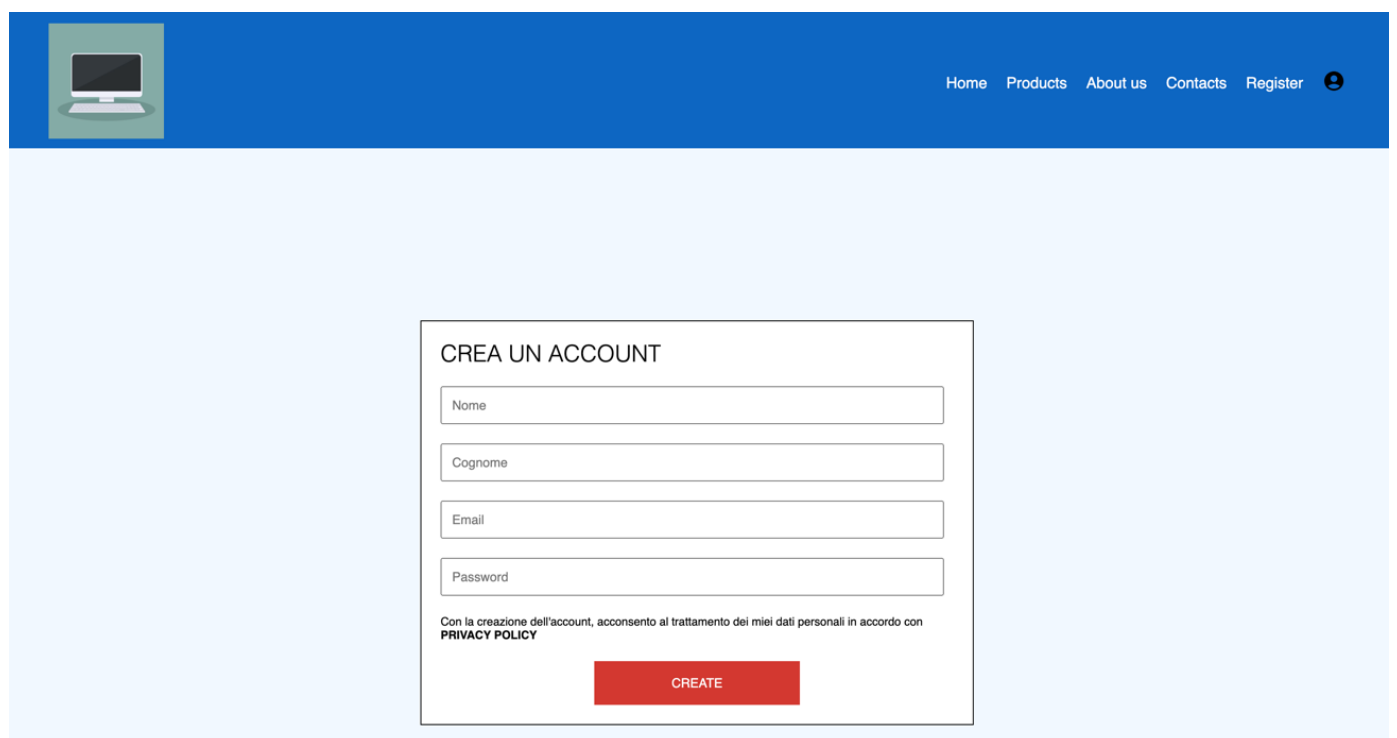
Password

LOGIN

[NON RICORDI LA PASSWORD?](#)

[CREA UN NUOVO ACCOUNT](#)

3.9.3 Creazione Account Page



The image shows a 'Create Account' page with a blue header. On the left of the header is a laptop icon. On the right are links: Home, Products, About us, Contacts, Register, and a user profile icon. The main content area is light blue and contains a white form titled 'CREA UN ACCOUNT'. The form includes input fields for 'Nome', 'Cognome', 'Email', and 'Password', followed by a red 'CREATE' button. At the bottom of the form, there is a line of text: 'Con la creazione dell'account, acconsento al trattamento dei miei dati personali in accordo con PRIVACY POLICY'.

CREA UN ACCOUNT

Nome

Cognome

Email

Password

Con la creazione dell'account, acconsento al trattamento dei miei dati personali in accordo con **PRIVACY POLICY**

CREATE