urCoach

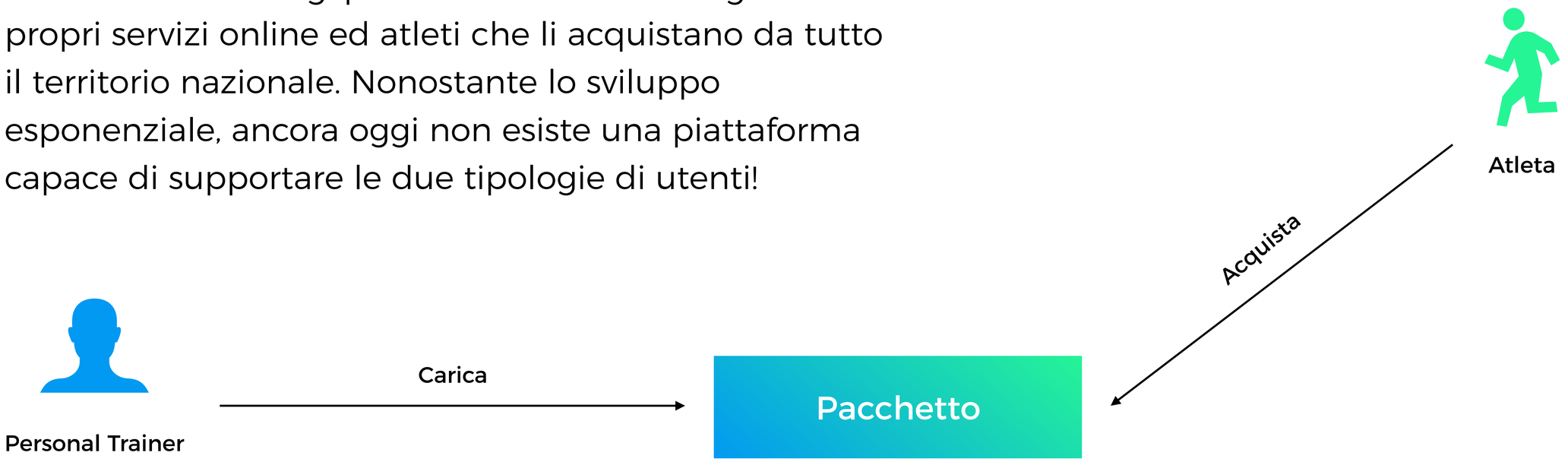
Ingegneria del Software AA 2019/2020

Salvatore Fasano 0512105068
Mariantonia Candela 0512105374

Requirement Analysis

Problem Statement

urCoach nasce come risposta al crescente fenomeno dell'online coaching: personal trainer che erogano i propri servizi online ed atleti che li acquistano da tutto il territorio nazionale. Nonostante lo sviluppo esponenziale, ancora oggi non esiste una piattaforma capace di supportare le due tipologie di utenti!



Analisi del sistema esistente

Non esiste un sistema informativo da analizzare, per cui analizziamo come la relazione tra i personal trainer e gli atleti viene gestita:

1. Primo contatto mediante social
2. Scambio di messaggi tramite social/email
3. Invio del materiale tramite email

Requisiti Funzionali

Oltre alle operazioni banali, come registrazione e login, il sistema deve garantire determinate funzionalità alle due tipologie di utenti:

Un personal trainer deve poter creare un pacchetto con tutte le informazioni correlate, gestire il proprio account e visualizzare le proprie vendite

Un atleta deve poter ricercare dei pacchetti attraverso il nome, filtrare all'interno di un elenco i pacchetti per una determinata categoria, un prezzo o un particolare personal trainer; deve poter inserire uno o più pacchetti nel carrello ed effettuare il checkout

Requisiti Funzionali

Naturalmente bisogna prevedere delle figure amministrative capace di controllare gli acquisti e le registrazioni dei personal trainer sull'eCommerce:

Un gestore ordini può controllare tutti gli acquisti effettuati tramite l'eCommerce

Un recruiter può decidere se accettare o meno un personal trainer che si è registrato

Requisiti Non Funzionali

Vogliamo che urCoach abbia diverse caratteristiche che lo rendano un prodotto competitivo sul mercato:

1. Deve essere semplice da utilizzare, un personal trainer non deve avere una laurea in informatica per poterlo utilizzare!
2. Deve essere sicuro, stabile, performante
3. Deve essere pronto a scalare in caso abbia un riscontro positivo sul mercato

System Design

Design Goal

Basandoci sui requisiti non funzionali formuliamo i design goal per urCoach, notiamo però che dobbiamo effettuare delle scelte.

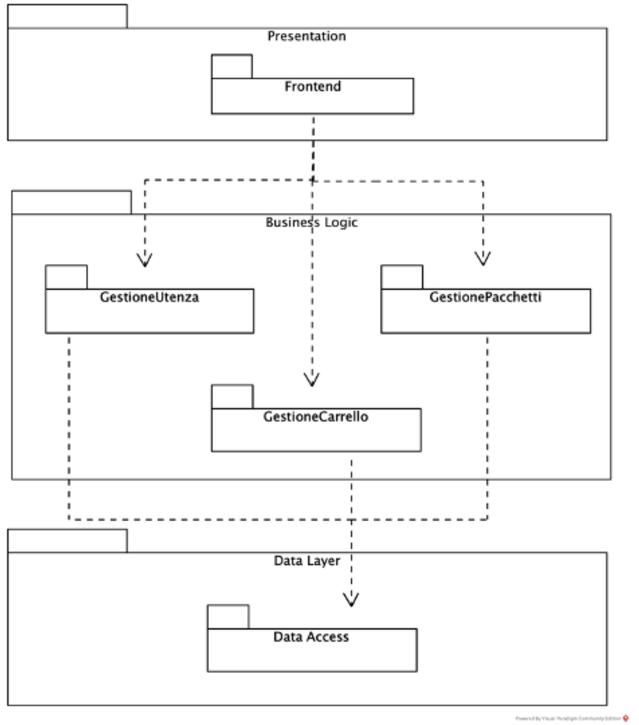
Tempo di rilascio VS Funzionalità

Prestazioni VS Costi

Architettura Proposta

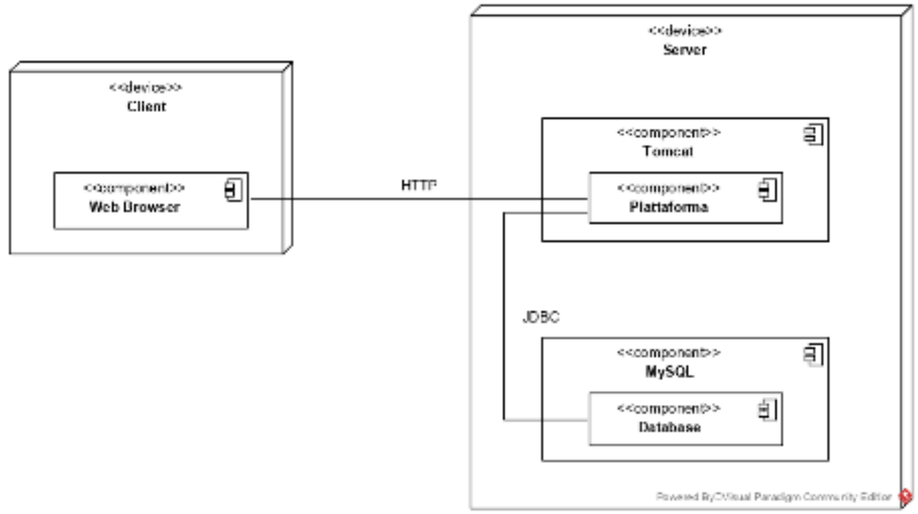
Optiamo per un’architettura MVC (Model/View/Control).
A destra possiamo osservare com’è stato decomposto il sistema:

- Livello di presentazione che racchiude la View
- Livello di logica che racchiude il Control
- Livello dati che si occupa del Model e della gestione della persistenza dei dati



HW/SW Mapping

urCoach sarà installabile su qualsiasi server capace di eseguire Java e MySQL. A causa del budget ridotto si decide di avere entrambi i componenti nello stesso server.



Dati Persistenti

Data la natura strutturata dei dati che andiamo ad utilizzare, abbiamo deciso di mantenere tutti dati persistenti all’interno di un database relazionale con MySQL.

Controllo d’Accesso

All’interno dell’eCommerce, come evidenziato in fase di analisi, esistono diversi ruoli ed ogni ruolo ha con se dei permessi che gli consentono di utilizzare determinate funzionalità piuttosto che altre.

Object Design

Componenti Off-the-shelf

Per rispettare il budget a disposizione e mantenere la data di consegna si è optato per l'utilizzo di componenti off-the-shelf gratuite ed open source.
Si evidenzia che non vengono utilizzate componenti del genere per il design dell'eCommerce in quanto si vuole un proprio branding.

Interfacce delle classi

Grazie all’utilizzo di Spring Boot e Spring JPA, possiamo essere sicuri che tutte le operazioni che accedono ai dati persistenti non contengano errori, non ci resta che definire le interfacce del classi di Service e delle classi che si occupano del control (in questo caso però, con pre e post condizioni)

Testing

Approccio al testing

Per assicurarci che il sistema sia privo di errori e rispetti i requisiti iniziali abbiamo condotto il testing in tre fasi:

1. Test di unità (JUnit/Mockito)
2. Test di integrazione (JUnit/Mockito)
3. Test di sistema (Selenium IDE)

Abbiamo configurato sulla nostra repo Github due componenti aggiuntivi: Travis CI e Codecov

- Travis CI (Continuos Integration) ci ha permesso di mantenere urCoach libero da bug con dei controlli ad ogni commi
- Codecov, eseguito insieme a Travis, notifica la branch coverage ad ogni commit

Test di unità

Sono stati effettuati ben **103** test di unità (white-box), che coprono il 100% dei branch a livello di control e di service. La percentuale di branch coverage totale è..

Test di integrazione

Ci sono circa 50 test di integrazione con cui abbiamo verificato, con una strategia bottom-up, che l’integrazione di varie componenti non introduca bug.

Test di sistema

Per i test di sistema abbiamo utilizzato Selenium IDE, ciò ci ha permesso di testare le interazioni dell'utente sul sistema finale e valutare se potrebbero condurre a situazioni errate.

urCoach

Ingegneria del Software AA 2019/2020

Salvatore Fasano 0512105068
Mariantonia Candela 0512105374