Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Laurea triennale in Informatica

Webapp "Biota": piattaforma web per il benessere gastrointestinale

Relatore
Armir Bujari
Università di Padova

Tutor aziendale
Simone Pozzobon

Laureando
Andrea Trevisin

Alla mia famiglia, che mi ha costantemente supportato in questo lungo percorso, e ai miei amici, con cui ho condiviso studio e vita quotidiana.

Abstract

Contents

Elenco delle figure

Elenco delle tabelle

Elenco degli acronimi

API . . . . . Application Programming Interface

AWS . . . . . Amazon Web Services

CoC . . . . . Convention over Configuration

DSL . . . . . Domain Specific Language

DRY . . . . . Don't Repeat Yourself

ERB . . . . . Embedded RuBy

HOTP . . . . . HMAC-based One Time Password

HTTP . . . . . HyperText Transfer Protocol

IDE . . . . . Integrated Development Environment

JSON . . . . . JavaScript Notation Object

MVC . . . . . Model View Controller

OTP . . . . . One Time Password

PDF . . . . . Portable Document Format

REST . . . . . Representational State Transfer

SDK . . . . . Software Development Kit

UML . . . . . Unified Modeling Language

URI . . . . . Uniform Resource Identifier

TOTP . . . . . Time-based One Time Password

1 Introduzione

Il presente documento espone il lavoro svolto dal laureando Andrea Trevisin durante il periodo di stage formativo presso Moku S.r.l. al fine di realizzare il *back end* dell'applicazione web "Biota", una piattaforma per la gestione di esami della flora gastrointestinale. Lo stage, della durata di 300 (trecento) ore, si è svolto tra il 15 maggio e il 17 luglio 2019 presso la sede di Roncade. Il progetto prevedeva il raggiungimento di diversi obiettivi, suddivisi in *milestones*, a partire dalla realizzazione di un *back end* completo delle funzionalità base della piattaforma, per poi implementare i requisiti opzionali e una suite di test completa e infine soddisfare alcuni requisiti desiderabili. Al termine della durata dello stage, la piattaforma è in stato di produzione e tutti gli obiettivi sono stati soddisfatti.

1.1 Azienda

Moku S.r.l. è una startup nata nel 2013, con sede a Roncade (TV), fondata su un progetto software omonimo per una piattaforma web mirata a facilitare il lavoro condiviso su documenti di vario tipo. Si è poi evoluta diventando una società di consulenza IT che realizza prodotti software su commissione. Il team di Moku usa metodologie agili, basate su Scrum, e lavora a stretto contatto con il cliente; questo permette di individuare facilmente e velocemente i requisiti del prodotto, creando prodotti di qualità che rispecchiano le esigenze del committente. I principali ambiti di sviluppo di Moku sono piattaforme web applicazioni

Android/IOS.

Figure 1.1: Logo di Moku S.r.l.

1.2 Processi aziendali

1.2.1 Modello di ciclo di vita del software

Il modello di ciclo di vita del software adottato dal team di Moku si basa sulla metodologia Scrum. Scrum è un *framework* agile per la gestione del ciclo di sviluppo del software, iterativo ed incrementale, creato e sviluppato da Ken Schwaber e Jeff Sutherland nel 1995; si propone di essere leggero, semplice da imparare ma difficile da padroneggiare. Scrum si presta bene alla modalità di lavoro di Moku, in quanto è ideato per team di dimensioni ridotte e permette di gestire progetti complessi offrendo la possibilità di reagire velocemente a cambiamenti. Una delle idee chiave è infatti la "volatilità dei requisiti", ovvero riconoscere che le esigenze dei clienti possano variare in corso d'opera, e che possano sorgere complicazioni non previste - specialmente quando si usano tecnologie all'avanguardia.

L'obiettivo principale è quindi la realizzazione di un prodotto software funzionante e soddisfacente a scapito di aspetti ritenuti non essenziali nell'immediato (e.g. una documentazione completa).

1.2.1.1 Principi della metodologia Scrum

Alla base del *framework* vi è la teoria dei controlli empirici dell'analisi strumentale e funzionale di processo, altrimenti nota come "empirismo", la quale afferma che la conoscenza deriva dall'esperienza e le decisioni si devono basare su ciò che si conosce. L'implementazione dei controlli empirici di processo si basa su tre concetti:

- **Trasparenza**: gli aspetti significativi del processo devono essere visibili ai responsabili del prodotto; la trasparenza richiede che tali aspetti siano definiti secondo uno standard comune in modo che gli osservatori condividano una comprensione comune di ciò che avviene.

- **Ispezione**: chi usa Scrum deve ispezionare spesso i prodotti della metodologia e il progresso verso l'obiettivo di uno *sprint* per individuare eventuali difformità; tuttavia tale processo non dovrebbe essere frequente al punto da rallentare il lavoro. Le ispezioni danno il massimo beneficio quando eseguite da ispettori qualificati.

- **Adattamento**: se un'ispezione determina che uno o più aspetti sono al di fuori di certi limiti, e che il prodotto risultante è inaccettabile, il processo o il prodotto del processo vanno adattati; l'adattamento deve avvenire il prima possibile per evitare ulteriori difformità.

1.2.1.2 Sprint

Scrum prevede di dividere il progetto in blocchi contigui di lavoro, denominati *sprint*, che prevedono un incremento del prodotto software rilasciabile al termine di ciascuno di essi. Uno *sprint* dura da 1 a 4 settimane, ed è preceduto da una riunione di pianificazione (*sprint planning*) in cui vengono discussi e definiti obiettivi e stimate le tempistiche. Gli obiettivi stabiliti per uno *sprint* possono variare solo allo *sprint planning* successivo; ogni giorno viene fatta una breve riunione, detta *daily scrum*, in cui viene brevemente discusso il lavoro da svolgere.

Nel corso di uno *sprint*, il team realizza incrementi completi del prodotto: l'insieme di funzionalità inserite in un dato *sprint* sono definite nel *product backlog*, una lista ordinata di requisiti del prodotto che copre l'intero progetto. I requisiti scelti per essere soddisfatti durante un determinato sprint vanno a costituire lo *sprint backlog*.

Gli *sprint* in Moku durano generalmente 10/15 giorni, seguiti da una fase di valutazione del progresso rispetto alle aspettative e di pianificazione dello *sprint* successivo, in cui viene stabilito il nuovo *sprint backlog*. Dato il disaccoppiamento tra *back end* e *front end* nel progetto, i requisiti erano separati in due *backlog differenti*.

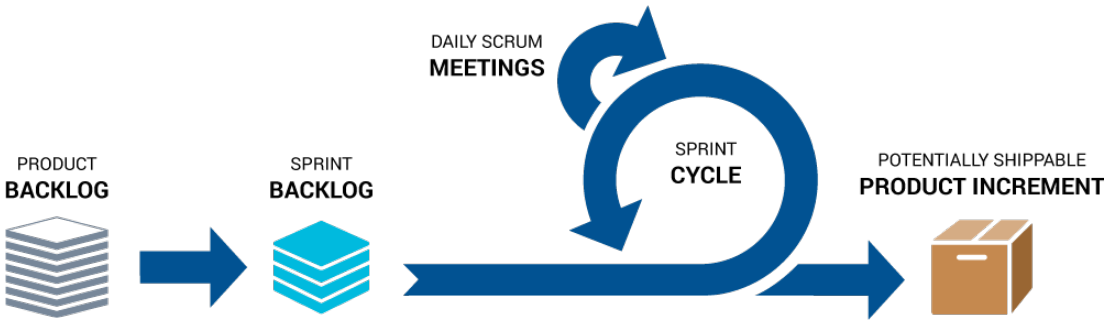

Figure 1.2: Rappresentazione grafica della metodologia Scrum

1.2.1.3 Ruoli di un team Scrum

Il *framework* Scrum prevede che i team siano auto-organizzati, ovvero in grado di scegliere autonomamente il modo migliore per portare a termine il lavoro, e interfunzionali, cioè che non dipendono da altri o da parte del team per realizzare il prodotto. I team Scrum lavorano in maniera iterativa ed incrementale, massimizzando le possibilità di feedback utile.

In un team Scrum sono presenti i seguenti ruoli:

- **Product Owner**: responsabile della massimizzazione del valore del prodotto, rappresenta la "voce" degli stakeholders; definisce gli *items* in base alle *user stories* del cliente, ne assegna la priorità e li aggiunge al *product backlog*.
- **Team di sviluppo**: responsabile della consegna del prodotto, con funzionalità incrementali e rilasciabile al termine di ogni *sprint*; generalmente composto da 3-9 persone con competenze interfunzionali, che realizzano il prodotto effettivo.
- **Scrum Master**: responsabile della rimozione degli ostacoli che intralciano il team di sviluppo nel raggiungimento degli obiettivi dello *sprint*; aiuta il team di sviluppo a comprendere e mettere in pratica la metodologia Scrum, assumendo un ruolo di "*servant/leader*".

Per la durata dello stage, nonostante le linee guida di Scrum lo sconsiglino, i ruoli di *Product Owner* e *Scrum Master* erano ricoperti entrambi dal tutor aziendale per praticità. Il team

di sviluppo, composto da due persone, includeva il laureando, il tutor aziendale (che interveniva solo ove fossero necessarie correzioni minori) e uno/due sviluppatori *front end* (a seconda della disponibilità).

1.2.2 Strumenti a supporto dei processi

Per quanto riguarda gli strumenti di supporto ai processi, Moku usa per la maggior parte prodotti appartenenti all'ecosistema Atlassian, ottenendo una maggiore integrazione tra gli strumenti.

1.2.2.1 Project management

Per quanto riguarda la gestione dei progetti, Moku fa affidamento a Jira, un prodotto proprietario di Atlassian. Jira è uno strumento estremamente versatile che offre funzioni di *issue tracking*, *project management* e *bug tracking*. Jira è inoltre fortemente orientato all'utilizzo della metodologia Scrum, permettendo di creare degli *sprint* e di gestire il rispettivo *backlog*.

- **Issue tracking**: in Jira è possibile creare *issues* di tipo *Task* per compiti da completare, *Story* per nuove (*user stories*) e *Feature/Improvement* per funzionalità o miglioramenti a funzionalità esistenti. Ogni *issue* può essere assegnata ad uno o più membri del team, che può aggiornarne il progresso, dividerla in *sub-task* e registrare le ore di lavoro.
- **Enterprise resource planning**: Jira mette a disposizione uno strumento di pianificazione della distribuzione delle risorse temporali ed economiche, sotto il nome di *Tempo*; esso permette di pianificare in anticipo il lavoro da dedicare ad una certa *issue* per ogni membro del team, nonché di tenere traccia dei budget assegnati al progetto.
- **Bug tracking**: in Jira avviene creando issue di tipo *Bug*.

Jira mette a disposizione un'interfaccia drag-and-drop, che ne rende l'uso veloce ed intuitivo.

1.2.2.2 Gestione del versionamento

Per la gestione del versionamento del prodotto software Moku usa *repository Git* sul servizio di hosting Bitbucket, parte dell'ecosistema Atlassian. Bitbucket offre la possibilità di creare gratuitamente *repository* private, risultando una scelta migliore per l'azienda. Queste vengono gestite localmente tramite il client dedicato Sourcetree, anch'esso proprietario di Atlassian (quindi perfettamente integrato con Bitbucket), il quale semplifica l'uso del paradigma *git-flow*, scelto come standard da Moku.

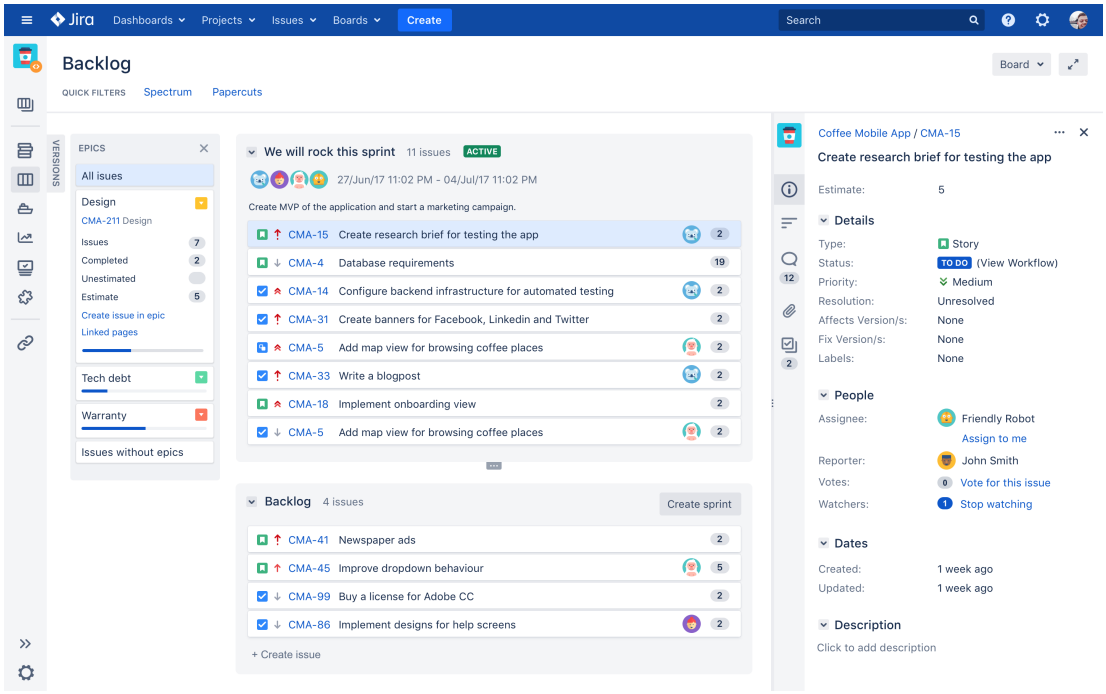

Figure 1.3: Interfaccia della sezione di issue tracking di Jira

GIT-FLOW Paradigma di versionamento utilizzato per il progetto, che prevede l'utilizzo sincronizzato di molteplici branch nella repository centrale del prodotto. Il branch **master** contiene l'ultima *release* del prodotto software, ovvero l'ultima versione stabile che include un certo numero di funzionalità testate. I *commit* a questo branch hanno sempre natura incrementale in quanto corrispondono al rilascio di una nuova versione del prodotto. Il branch **develop** invece contiene il prodotto software in lavorazione ed è soggetto a *commit* più frequenti, corrispondenti allo sviluppo di una nuova funzionalità; in ogni caso, in questo branch si trova un prodotto funzionale ma incompleto. Infine è prevista la creazione di vari branch temporanei, uno per cisacuna *feature* in via di sviluppo. Per aggiungere una *feature* al prodotto, viene effettuato un *fork* del branch **develop** e creato un branch **feature** apposito: al completamento dello sviluppo della funzionalità, viene effettuato il *merge* con il branch di sviluppo. È possibile correggere un errore propagatosi al branch principale con un *hotfix* un *commit* di emergenza. Nel progetto è stato inoltre utilizzato un terzo branch, **staging**, per testare l'integrazione tra *back end* e *front end* prima del rilascio della nuova versione.

1.2.2.3 Gestione delle comunicazioni

A livello aziendale La corrispondenza riguardante l'avvio del progetto di stage e la collaborazione con il cliente, così come altre comunicazioni a livello aziendale, sono avvenute tramite posta elettronica.

Nel team Per facilitare la comunicazione tra i membri di ogni team di sviluppo, nonché per comunicazioni meno importanti, Moku fa uso di Slack. Slack è un applicativo *cloud-based* di messaggistica e file-sharing, in cui i messaggi vengono inviati in "canali": questo permette di dividere le comunicazioni per progetto, facilitando anche la ricerca di informazioni rilevanti precedentemente inviate.

1.3 Organizzazione del testo

1.4 Convenzioni tipografiche

If it looks like a duck, and quacks like a duck, we have at least to consider the possibility that we have a small aquatic bird of the family Anatidae on our hands.

Douglas Adams

2 Progetto di stage

2.1 Descrizione del progetto

Il progetto di stage prevedeva lo sviluppo della parte *back end* dell'applicazione web "Biota". "Biota" è una piattaforma per la gestione di esami della flora gastrointestinale, che permette ai clienti di una farmacia di richiedere un esame mediante la compilazione di un questionario; un campione di materia fecale viene quindi inviato al laboratorio per le analisi, al cui termine viene fornito un referto sulla base del quale vengono consigliati prodotti atti a migliorare il benessere del cliente. Il *back end* comprende un pannello amministrativo per la gestione dei record del database, permettendone creazione, distruzione e modifica nonchè consentendo operazioni aggiuntive quali il caricamento e lo scaricamento di file per i record designati.

2.1.1 Contesto di sviluppo

La piattaforma fa parte di un ecosistema già esistente di servizi per la salute, fruibili presso farmacie affiliate. Esiste, quindi, come parte di un insieme di applicativi paralleli e ne condivide il database e parte della codifica: questo documento si limiterà a descrivere i moduli, le classi e le funzionalità realizzate o adattate per l'implementazione dellla piattaforma "Biota". L'applicativo si compone di una parte *back end* (in esecuzione su un server dedicato) e una parte *front end* (eseguita all'interno del browser dell'utente) sviluppate separatamente e integrate grazie all'uso di API specifiche; obiettivo dello stage è stata la realizzazione del primo,

in modo funzionale all'utilizzo con il secondo. La tecnologia principale, descritta più avanti nel documento, è Ruby on Rails 5 (*framework* per lo sviluppo di applicativi web).

2.1.2 Caratteristiche degli utenti

Trattandosi di una piattaforma commerciale, ne è previsto l'utilizzo di utenti autorizzati con diversi: farmacisti, gastroenterologi e addetti del laboratorio. Tutte le categorie di utenti autorizzati eseguono il *login* nella piattaforma allo stesso modo, ma ad ognuno viene mostrata un'interfaccia con diverse funzionalità. Il cliente non fa parte degli utenti autorizzati in quanto non può accedere direttamente alla piattaforma, e tutte le operazioni che lo coinvolgono sono supervisionate dal farmacista. Per quanto riguarda il pannello amministrativo, ne è previsto l'utilizzo da parte di utenti autorizzati registrati separatamente; tali amministratori fanno parte della società che gestisce la piattaforma.

2.1.3 Contesto di utilizzo

"Biota" prevede l'utilizzo da parte di utenti registrati alla piattaforma, che possono essere farmacisti, clienti o gastroenterologi affiliati. A questi viene offerta un'interfaccia grafica facente parte del *front end*. Per interazioni con servizi esterni sono inoltre disponibili delle API REST, utilizzate principalmente per l'integrazione con il gestionale del laboratorio di analisi.

Il flusso operativo previsto è il seguente: il cliente richiede al farmacista di effettuare un esame, e inserisce i propri dati registrandosi alla piattaforma; il farmacista quindi ottiene il consenso al trattamento dei dati e fa compilare un breve questionario al cliente, consegnando un kit per il prelievo del campione di materia fecale. Questo viene poi restituito, registrato al cliente ed inviato al laboratorio previa prenotazione della spedizione: grazie alla corrispondenza univoca tra il codice identificativo del campione e la sessione associata all'esame, è garantito l'anonimato del cliente. Il laboratorio, ricevuto il campione, ne effettua il check-in nel proprio gestionale (aggiornandone automaticamente lo stato nella piattaforma) e procede alle analisi necessarie. Viene quindi inviato una bozza di referto insieme a dei dettagli aggiuntivi, che viene reso disponibile nella piattaforma al gastroenterologo affiliato che aggiunge una sua valutazione e dieta consigliata. Dopo un'ulteriore controllo da parte del laboratorio, il referto personalizzato viene generato reso disponibile al cliente, concludendo la procedura.

L'accesso al pannello amministrativo consente di creare, distruggere e modificare alcune categorie di record del database, come ad esempio le varie tipologie di utenti o le farmacie af-

filiate. Esso viene utilizzato principalmente per l'inserimento di nuovi utenti autorizzati, il recupero di informazioni rilevanti quali il consenso al trattamento dei dati, e per la risoluzione di problematiche grazie all'accesso privilegiato al database.

2.2 Tecnologie utilizzate

2.2.1 Ruby

Ruby è un linguaggio di programmazione ad alto livello, interpretato, e orientato agli oggetti con paradigma puro: ogni componente del linguaggio è trattato come un oggetto.

Alcune tra le caratteristiche più rilevanti di Ruby sono la presenza di tipizzazione dinamica, *garbage collector* e *duck typing* ("Se sembra un'anatra, nuota come un'anatra e starnazza come un'anatra, allora probabilmente è un'anatra."), ovvero considera l'insieme dei metodi di un oggetto anziché il suo tipo per decidere se è valido a *run-time* .

Si tratta un linguaggio molto flessibile che offre grande libertà allo sviluppatore e supporta pratiche come il *monkey patching* (ridefinire una classe in un punto diverso dalla definizione originale) e la ridefinizione di metodi a *run-time*. Esiste una grande varietà di programmi e librerie Ruby, noti come "gemme", il cui utilizzo verrà discusso in seguito.

2.2.2 Gemme

Le gemme sono librerie e programmi Ruby distribuiti sotto forma di pacchetti in modo non dissimile dai moduli in Node.js: l'installazione è gestita dal *package manager* RubyGem, individualmente tramite terminale (con il comando `gem install nomegemma`) oppure di gruppo specificando le gemme desiderate nel file `Gemfile`, che viene automaticamente letto da RubyGem con l'esecuzione del comando `bundle install`.

2.2.2.1 HexaPDF

Libreria che permette interazioni ad alto e basso livello con file PDF: è possibile leggere e modificare il codice sorgente di un documento, oppure sfruttare i *wrappers* presenti per effettuare operazioni ad alto livello come l'inserimento di immagini o figure. Utilizzata per la generazione dinamica di referti.

2.2.2.2 ROTP

Acronimo di "Ruby One Time Password", questa gemma permette di generare e verificare codici HOTP e TOTP in accordo agli standard RFC 4426 [1] e RFC 6238 [2]. ROTP è inoltre compatibile con Google Authenticator su dispositivi Android e iOS. La gemma è stata sfruttata per verificare il consenso al trattamento dei dati mediante codici TOTP inviati per SMS.

2.2.2.3 AWS SDK for Ruby

Insieme di gemme che facilitano l'interazione con i servizi web di Amazon, fornendo classi e metodi ad-hoc; l'SDK è suddiviso in moduli specifiche per ogni servizio, permettendo di scegliere quali gemme usare a seconda delle necessità, e di aggiornare le gemme utilizzate in modo indipendente. In particolare è stata utilizzata la gemma `aws-sdk-sns` per l'invio di SMS tramite Amazon Simple Notification Service.

2.2.3 Ruby on Rails

Per la realizzazione del progetto Moku ha scelto di usare Ruby on Rails 5 (noto anche come Rails), un *framework open source* per applicativi web realizzato in Ruby e utilizzato da celebri siti come GitHub (servizio di hosting per *repository* Git), Twitch (servizio di *live streaming*) e SoundCloud (piattaforma di distribuzione musicale). Rails è stato scelto per la caratteristica di velocizzare notevolmente lo sviluppo di nuovi applicativi, rimuovendo le parti "ripetitive": ad esempio offrendo alias concisi per operazioni di base (e.g. iterazioni su collezioni di oggetti, strutture if/else) che riducono la verbosità del codice.

I principi cardine di Rails sono due:

- **Do not Repeat Yourself**: il principio DRY sostiene che vadano evitate tutte le forme di ripetizione e ridondanza logica nell'implementazione del software; ad esempio, non è necessario specificare le colonne della tabella del database nella definizione di una classe, in quanto Rails recupera automaticamente tale informazione.

- **Convention over Configuration**: il principio CoC sostiene che il programmatore dovrebbe esplicitare solo le parti "non convenzionali" del codice; ad esempio in Rails esiste per convenzione una corrispondenza tra il nome di una classe e il nome di una tabella del database, quindi essa non va specificata.

[1] http://tools.ietf.org/html/rfc4226
[2] http://tools.ietf.org/html/rfc6238

2.2.3.1 Integrazione tra i componenti

Rails è un *framework full-stack*, ovvero offre tutti i componenti richiesti per lo sviluppo di un applicativo web, nativamente integrati tra di loro: tramite l'uso di script per la creazione di file, detti *generators*, è possibile creare contemporaneamente sia una tabella del database che la classe corrispondente; essi sono automaticamente associati tramite convenzioni di nomenclatura.

Il database, indipendentemente dall'implementazione, può essere modificato tramite *migrations*, istruzioni per la modifica dello schema da parte di Rails. Le *migrations* sono scritte in un DSL in Ruby e versionate automaticamente, permettendo di effettuare un *rollback* ad una versione precedente dello schema.

2.2.3.2 Pattern architetturale

Gli applicativi realizzati in Rails seguono necessariamente un pattern *MVC*.

Model Un *model* è una classe associata ad una tabella del database secondo uno standard convenzionale: una tabella corrisponde una classe, le colonne sono convertite in attributi e le righe rappresentate come istanze della classe. I *models* possono venire generati automaticamente dalla definizione della tabella. Oltre ai normali vincoli di database, in Rails è possibile definire ulteriori controlli sui valori in database direttamente nel *model*. La filosofia di Rails prevede che essi contengano la quasi interezza della *business logic*.

Controller I *controllers* sono componenti che rispondo alle richieste del server web, determinando quale *view* caricare; essi possono anche interrogare i *models* per mostrare informazioni aggiuntive, e rendere disponibili "azioni" per fare richieste al server. I controller sono resi disponibili mediante il file di *routing* `routes.rb`, in cui vengono associati a delle specifiche richieste; Rails incoraggia gli sviluppatori all'uso di *RESTful routes*, che includono azioni come "index", "new", "show", "edit".

View Di default le *views* sono file con estensione `.erb` contenenti codice HTML misto a codice Ruby, che vengono elaborati a *run-time* permettendo una visualizzazione dinamica delle informazioni. In alternativa, per esempio nell'implementazione di una RESTful API, una *view*può essere un file JSON contenente il *body* di una risposta.

2.2.3.3 Devise

Framework di autenticazione utenti per Rails. Si tratta di una gemma altamente modulare e flessibile, composta 10 sotto-moduli, ognuno dei quali implementa una data funzionalità, permettendo di "comporre" a proprio piacimento un sistema di autenticazione con le caratteristiche desiderate (e.g. recupero password, tracciamento di orari e indirizzi IP per ogni accesso e validazione tramite e-mail). Devise è stato utilizzato per l'autenticazione sulla piattaforma da parte degli utenti registrati.

2.2.3.4 ActiveAdmin

ActiveAdmin è un *framework* per la generazione di interfacce amministrative per applicativi web realizzati con Rails. ActiveAdmin astrae pattern ricorrenti per automatizzare la generazione di elementi comuni dell'interfaccia, ad esempio operazioni quali la creazione, visualizzazione o modifica di oggetti appartenenti a uno o più *models*. ActiveAdmin si integra con la configurazione presente di Devise per gestire l'autenticazione. In ActiveAdmin viene fornita un'interfaccia di default pienamente configurabile: è possibile aggiungere un pannello relativo ad una tabella particolare del database, applicarvi filtri, e decidere quali campi rendere disponibili alle operazioni di visualizzazione e modifica.

2.2.4 RubyMine

Ambiente di sviluppo integrato multipiattaforma, progettato specificamente per Ruby on Rails e realizzato da JetBrains. RubyMine mette a disposizione vari strumenti per facilitare lo sviluppo, tra cui completamento automatico del codice, linting avanzato con calcolo della complessità ciclomatica, e creazione guidata di *migrations* e *generators*. Altre caratteristiche che hanno portato a sceglierlo come IDE di riferimento sono la possibilità di testare ed eseguire l'applicativo in locale, il controllo di versione integrato (comodo per cambiare il *branch* corrente senza dover utilizzare Sourcetree) e lo strumento di *refactoring* che rispetta le convenzioni di Rails (e.g. modificando il nome di un *model*, i rispettivi *controller* e *view* vengono aggiornati.)

2.2.5 GraphQL

Linguaggio di query *open source*, sviluppato da Facebook nel 2012 e disponibile al pubblico dal 2015. La sintassi di GraphQL è molto simile a quella del formato JSON, pensata per una

lettura più facile da parte di operatori umani. La particolarità di GraphQL è che include anche il *runtime system* e il sistema dei tipi, quindi non dipende dalla specifica implementazione del database. I tipi sono definiti dallo sviluppatore e vengono utilizzati da GraphQL per validare le richieste e respingere query errate.

GraphQL offre due tipi di operazioni: query, semplici interrogazioni al database, e *mutations*, ovvero operazioni di modifica del database o interazioni particolari (autenticazione, esecuzione di un comando).

Il *runtime system* di GraphQL, eseguito sul server, è responsabile della validazione delle richieste e della serializzazione delle risposte in formato JSON. Sono disponibili librerie per creare API in vari linguaggi di programmazione, tra cui Ruby.

Nel progetto GraphQL è stato usato per implementare l'API di comunicazione con il *front end* della piattaforma, mettendo a disposizione varie query rilevanti nonché *mutations* per autenticazione, interazione con il database e richieste di elaborazione al *back end*.

2.2.5.1 Altair

Ambiente di sviluppo integrato *open source* e multipiattaforma per GraphQL. Altair fornisce una semplice interfaccia per testare API GraphQL, mettendo a disposizione funzioni utili quali formattazione automatica,generazione automatica della documentazione del sistema dei tipi e aggiunta di *headers* alla richiesta. Altair è stato utilizzato per testare le API GraphQL prima di renderle disponibili al *front end*.

2.2.6 HTTP

Noto protocollo a livello applicativo per la trasmissione di informazioni in una rete. Il protocollo HTTP prevede un'architettura di tipo *client/server*, in cui il *client* esegue una richiesta e il *server* la elabora, fornendo una risposta adeguata.

Una richiesta HTTP si compone di quattro parti:

- una "riga di richiesta" che specifica il tipo di operazione e l'URI dell'oggetto della richiesta;
- una "sezione *header*" in cui vengono fornite informazioni aggiuntive (e.g. credenziali di autenticazione);
- una riga vuota che contiene i due caratteri *carriage return* e *line feed*;
- un *body*, ovvero il corpo del messaggio.

Il messaggio di risposta segue una struttura simile, con l'eccezione della riga di richiesta che viene sostituita da una "riga di stato",contenente informazioni sul risultato della richiesta sotto forma di codici standard.

Il protocollo HTTP è largamente utilizzato per l'implementazione di RESTful API; per quanto riguarda il progetto, è stato utilizzato per realizzare l'API che si sarebbe interfacciata con il software gestionale del laboratorio di analisi, esponendo determinati *endpoint* per le richieste.

2.2.6.1 Insomnia

Client *open source* e multipiattaforma per API GraphQL e REST, consente di gestire facilmente lo sviluppo e il testing di richieste HTTP e GraphQL; offre funzionalità di formattazione automatica, variabili d'ambiente e supporto a vari tipi di autenticazione. Insomnia è stato utilizzato per testare gli endpoint delle API per il laboratorio di analisi, nonché per sviluppare un client per le API del corriere SDA.

2.2.7 PostgreSQL

This is some random quote to start off the chapter.

Firstname lastname

3 Analisi dei requisiti

3.1 Definizione dei requisiti

Nella seguente sezione verranno esposti in via generale i requisiti che l'applicativo realizzato doveva soddisfare. È importante notare che, data la natura agile del metodo di lavoro impiegato dall'azienda, gli obiettivi presentati nel piano di lavoto sono stati soggetti a modifiche e aggiunte di lieve e media entità; per completezza e possibilità di confronto, di seguito verranno esposti sia i requisiti iniziali che quelli elaborati a fronte dell'interazione con il cliente.

3.1.1 Notazione

Si farà riferimento a requisiti ed obiettivi secondo le rispettive notazioni:

- **O** indica un requisito obbligatorio, vincolante in quanto obiettivo primario richiesto dal committente.
- **D** indica un requisito desiderabile, non vincolante o strettamente necessario, ma dal riconoscibile valore aggiunto.
- **F** indica un requisito facoltativo, rappresentante un valore aggiunto non necessariamente competitivo.

3.1.2 Piano di lavoro

Il piano di lavoro, stilato in collaborazione con il relatore ed il tutor aziendale nella settimana precedente all'inizio dello stage, espone una serie di requisiti emersi dalle esigenze del cliente emerse in fase di avvio del progetto.

3.1.2.1 Requisiti

Obbligatori

- O01: Analisi ed implementazione API REST laboratorio
- O02: Analisi ed implementazione API GraphQL per il flusso operativo base

Desiderabili

- D01: Analisi ed implementazione di test dei sistemi di pagamento e fatturazione
- D02: Suite di testing del software prodotto
- D03: Documentazione completa

Facoltativi

- F01: Ulteriori modifiche all'applicazione che esulano da quando riportato nel piano di lavoro

3.1.3 Modifiche al piano

Durante il corso dello stage, nello svolgimento del ruolo di Product Owner da parte del tutor aziendale, ci sono state varie interazioni con il cliente che hanno portato a ridefinire alcuni requisiti desiderabili e a specificare in modo più preciso i requisiti facoltativi. Tali modifiche sono sempre state fatte in accordo alle possibilità di soddisfazione entro i tempi previsti, e tenendo in mente che la piattaforma sarebbe entrata in produzione al completamento dellle funzionalità per l'esecuzione flusso di attività base.

3.1.3.1 Requisiti rivisti

Obbligatori

- O01: Analisi ed implementazione API REST laboratorio
- O02: Analisi ed implementazione API GraphQL per il flusso operativo base
- O03: Implementazione delle strutture per il questionario

Desiderabili

- D01: Generazione PDF con informazioni specifiche
- D02: Suite di testing del software prodotto
- D03: Documentazione completa
- D04: Registrazione del consenso al trattamento dei dati

Facoltativi

- F01: Autenticazione mediante HMAC
- F02: Ulteriori modifiche all'applicazione

3.2 Casi d'uso

Di seguito verranno descritti i casi d'uso dell'applicazione, per quanto riguarda le funzionalità da me progettate; non verranno quindi tenute in considerazione eventuali funzionalità preesistenti che fossero parte dell'ecosistema a cui appartiene la piattaforma.

I casi d'uso verranno inoltre considerati per quanto concerne il *back end* dell'applicazione, ovvero sulla base delle funzionalità realizzate per essere utilizzate dal *front end*: eventuali aggiunte, quali la visualizzazione di messaggi di errore dettagliati o reindirizzamenti a pagine differenti della piattaforma web, verranno ignorate.

I casi d'uso verranno descritti mediante dei diagrammi UML che rappresentano le possibili interazioni tra gli **attori** ed il **sistema** dal punto di vista dei primi. È importante notare che il sistema, ovvero il *back end* sviluppato, non viene utilizzato direttamente dagli utenti finali ma risponde a richieste di servizi esterni come il *front end* (con l'eccezione degli utenti amministrativi, a cui viene fornita un'interfaccia di interazione ad-hoc): i casi d'uso verranno dunque trattati di conseguenza.

3.2.1 Attori

- **Utente non autenticato**: un utente che deve effettuare l'autenticazione per accedere alle funzionalità disponibili al suo ruolo;
- **Amministratore**: un utente autenticato con privilegi di accesso al pannello amministrativo.
- **Front end**: una istanza del *front end* della piattaforma;
- **Gestionale laboratorio**: il software gestionale del laboratorio;

3.2.2 Struttura

Per ogni caso d'uso, identificato da un codice univoco, vengono specificati gli attori coinvolti, lo scopo da raggiungere, le azioni previste e le condizioni del sistema prima e dopo tali azioni. I casi d'uso verranno descritti dalla seguente struttura:

- Codice identificativo: **UC {codice_padre}.{codice_figlio}**
  - **UC** indica che si tratta di un caso d'uso;
  - **codice_padre** identifica il caso d'uso principale a cui si fa riferimento;
  - **codice_figlio** identifica il sottocaso specifico;
- Titolo
- Diagramma UML
- Attori
- Attori secondari, se presenti
- Scopo e descrizione
- Precondizioni
- Scenario principale
- Postcondizioni
- Inclusioni (se presenti)
- Estensioni (se presenti)

3.2.3 Casi d'uso generali

Per una miglior suddivisione, le operazioni che avvengono sotto condizioni molto simili (principalmente quelle che possono essere richieste da un utente autenticato con un determinato ruolo) saranno raggruppate in dei casi d'uso di alto livello, d'ora in poi "casi d'uso generali", identificati dalla lettera **G** prefissa al loro codice numerico. Tali casi d'uso servono solo per una miglior comprensione, quindi il loro codice non verrà considerato come **codice_padre** nell'identificazione dei casi d'uso raggruppati.

3.3 UC1: Autenticazione amministratore

Attori: Utente non autenticato.
Scopo e descrizione: L'attore vuole effettuare la procedura di autenticazione al pannello di amministrazione.
Precondizioni: L'applicativo è stato avviato con successo e l'attore ha accesso alla pagina di autenticazione.
Scenario principale:

- L'attore inserisce un indirizzo email valido (UC 1.1);
- L'attore inserisce la password associata a tale indirizzo email (UC 1.2);
- L'attore clicca sul pulsante di *login*.

Postcondizioni: L'attore viene autenticato dal sistema e ha accesso al pannello di amministrazione.
Estensioni: L'attore non viene autenticato e visualizza un messaggio di errore (UC2).

3.3.1 UC1.1: Inserimento email

Attori: Utente non autenticato.
Scopo e descrizione: L'attore vuole inserire un indirizzo email per l'autenticazione al pannello di amministrazione.
Precondizioni: L'attore ha accesso alla pagina di autenticazione.
Scenario principale: L'attore inserisce un indirizzo mail nell'apposito campo.
Postcondizioni: L'attore ha inserito un indirizzo email valido.

3.3.2 UC1.2: Inserimento password

Attori: Utente non autenticato.
Scopo e descrizione: L'attore vuole inserire una password per l'autenticazione al pannello di amministrazione.
Precondizioni: L'attore ha accesso alla pagina di autenticazione.
Scenario principale: L'attore inserisce una password nel campo apposito.
Postcondizioni: L'attore ha inserito una password corretta.

3.4 UC2: Errore autenticazione amministratore

Attori: Utente non autenticato.
Scopo e descrizione: L'attore vuole effettuare la procedura di autenticazione al pannello di amministrazione.
Precondizioni: L'attore ha inserito le proprie credenziali negli appositi campi.
Scenario principale:

- L'attore clicca sul pulsante di *login*;
- L'attore visualizza un messaggio di errore che lo informa che le credenziali non sono corrette.

Postcondizioni: L'attore non viene autenticato dal sistema e rimane nella pagina di autenticazione.

3.5 UC3: Autenticazione front end

Attori: Front end.
Scopo e descrizione: L'attore vuole effettuare la procedura di autenticazione per un utente della piattaforma.
Precondizioni: L'applicativo è stato avviato con successo e l'attore è collegato correttamente al sistema.
Scenario principale: L'attore invia una richiesta di autenticazione contenente le credenziali di accesso di un utente autorizzato.
Postcondizioni: L'utente specificato dall'attore viene autenticato, e l'attore riceve in risposta i parametri della relativa sessione attiva.

Estensioni: L'utente specificato dall'attore non viene riconosciuto, e l'attore riceve in risposta un messaggio di errore che lo informa che le credenziali non sono corrette (UC5).

3.6 UC4: Reset password per accesso alla piattaforma

Attori: Front end.
Scopo e descrizione: L'attore vuole effettuare la procedura di reset della password per un utente della piattaforma.
Precondizioni: L'applicativo è stato avviato con successo e l'attore è collegato correttamente al sistema.
Scenario principale: L'attore invia una richiesta di reset della password di accesso di un utente autorizzato.
Postcondizioni: È stata generata ed inviata all'utente una nuova password,e l'attore riceve una risposta di conferma del successo dell'operazione.

3.7 UC5: Errore di autenticazione front end

Attori: Front end.
Scopo e descrizione: L'attore vuole effettuare la procedura di autenticazione per un utente della piattaforma.
Precondizioni: L'attore ha inviato una richiesta di autenticazione al sistema.
Scenario principale:

- L'attore è in attesa di una risposta dal sistema;
- L'attore riceve in risposta un messaggio di errore che lo informa che le credenziali non sono corrette.

Postcondizioni: L'utente specificato dall'attore non viene riconosciuto e non riceve i dati di una sessione valida.

3.8 UC G1: Operazioni amministrative

Questo caso d'uso generale riassume le operazioni disponibili ad un utente autorizzato che abbia eseguito l'accesso al pannello di amministrazione. La maggior parte delle operazioni sui

record sono identiche tra loro, e verranno quindi riportate in modo generalizzato; eventuali operazioni specifiche verranno contestualizzate nel proprio caso d'uso. **Attori:** Amministratore.
Scopo e descrizione: L'attore vuole eseguire operazioni di amministrazione della piattaforma.
Precondizioni: L'attore ha ottenuto l'accesso al pannello di amministrazione.
Scenario principale:

- L'attore visualizza e gestisce una categoria di record (UC6);
- L'attore visualizza la *dashboard* (UC7).
- L'attore effettua la disconnessione dal pannello di amministrazione (UC8).

Postcondizioni: L'attore ha correttamente portato a termine le operazioni desiderate.

3.9 UC6: Gestione categoria di record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole gestire una particolare categoria di record.
Precondizioni: L'attore è nella *dashboard* del pannello di amministrazione.
Scenario principale:

- L'attore seleziona una scheda associato alla categoria desiderata e visualizza la tabella dei record.
- L'attore modifica l'ordine della tabella dei record (UC6.1);
- L'attore applica un filtro alla tabella dei record (UC6.2);
- L'attore rimuove un filtro dalla tabella dei record (UC6.3);
- L'attore aggiunge un nuovo record (UC6.4);
- L'attore visualizza un record particolare (UC6.5);
- L'attore modifica un record particolare (UC6.6);
- L'attore elimina un record particolare (UC6.7);
- L'attore gestisce un utente (UC6.10).

- L'attore gestisce un cliente (UC6.11).

Postcondizioni: L'attore ha eseguito tutte le operazioni di gestione della categoria di record selezionata con successo.

3.9.1 UC6.1: Modifica ordine della tabella dei record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole cambiare l'ordine della tabella dei record secondo un certo campo.
Precondizioni: L'attore sta visualizzando una tabella di record.
Scenario principale: L'attore seleziona la colonna del campo in base al quale ordinare la tabella. **Postcondizioni:** L'attore ha riordinato la tabella secondo il campo desiderato.

3.9.2 UC6.2: Applicazione filtro alla tabella dei record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole filtrare alcune righe della tabella dei record.
Precondizioni: L'attore sta visualizzando una tabella di record.
Scenario principale:

- L'attore seleziona l'icona del filtro;
- L'attore imposta il filtro desiderato;
- L'attore clicca sul pulsante"Filtra".

Postcondizioni: L'attore ha applicato il filtro desiderato alla tabella dei record.

3.9.3 UC6.3: Rimozione filtro alla tabella dei record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole rimuovere un filtro dalla tabella dei record.
Precondizioni: L'attore sta visualizzando una tabella di record con un filtro attivo.
Scenario principale:

- L'attore seleziona l'icona del filtro;
- L'attore clicca sul pulsante "Rimuovi filtri".

Postcondizioni: L'attore ha rimosso il filtro desiderato alla tabella dei record.

3.9.4 UC6.4: Aggiunta di un nuovo record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole aggiungere un record alla tabella.
Precondizioni: L'attore sta visualizzando una tabella di record.
Scenario principale:

- L'attore clicca sul pulsante "Aggiungi **{record}**", dove **{record}** è il nome della categoria selezionata;
- L'attore viene reindirizzato al *form* di creazione del record;
- L'attore riempe gli appositi campi con i dati desiderati;
- L'attore clicca sul pulsante "Crea **{record}**".

Postcondizioni: L'attore ha creato con successo un nuovo record.
Estensioni: L'attore annulla la creazione di un nuovo record (UC6.9).

3.9.5 UC6.5: Visualizzazione di un record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole visualizzare un record della tabella.
Precondizioni: L'attore sta visualizzando una tabella do record.
Scenario principale:

- L'attore clicca sul pulsante "Mostra" sulla riga corrispondente al record desiderato;
- L'attore visualizza i dati relativi al record;
- L'attore modifica il record;
- L'attore elimina il record.

Postcondizioni: L'attore ha eseguito tutte le operazioni di gestione della categoria di record selezionata con successo.

3.9.6 UC6.6: Modifica di un record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole modificare un record della tabella.
Precondizioni: L'attore sta visualizzando una tabella di record.
Scenario principale:

- L'attore clicca sul pulsante "Modifica" sulla riga corrispondente al record desiderato;
- L'attore viene reindirizzato al *form* di modifica del record;
- L'attore modifica il contenuto dei campi disponibili;
- L'attore clicca sul pulsante "Aggiorna **{record}**", dove **{record}** è il nome della categoria selezionata.

Scenario alternativo:

- L'attore visualizza il record desiderato (UC6.5);
- L'attore clicca sul pulsante "Modifica **{record}**", dove **{record}** è il nome della categoria selezionata;
- L'attore viene reindirizzato al *form* di modifica del record;
- L'attore modifica il contenuto dei campi disponibili;
- L'attore clicca sul pulsante "Aggiorna **{record}**";

Postcondizioni: L'attore ha correttamente aggiornato il record con i dati desiderati. **Estensioni:**

- L'attore annulla la modifica del record (UC6.9);
- L'attore commette un errore nela modifica del record (UC6.9).

3.9.7 UC6.7: Eliminazione di un record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole eliminare un record della tabella.
Precondizioni: L'attore sta visualizzando una tabella di record.
Scenario principale:

- L'attore clicca sul pulsante "Rimuovi" sulla riga corrispondente al record desiderato;
- L'attore visualizza un *pop-up* di conferma dell'operazione;
- L'attore clicca sul pulsante "Ok" per confermare l'operazione.

Scenario alternativo:

- L'attore visualizza il record desiderato (UC6.5);
- L'attore clicca sul pulsante "Rimuovi **{record}**", dove **{record}** è il nome della categoria selezionata;
- L'attore visualizza un *pop-up* di conferma dell'operazione;
- L'attore clicca sul pulsante "Ok" per confermare l'operazione.

Postcondizioni: L'attore ha correttamente rimosso il record. **Estensioni:** L'attore annulla la rimozione del record (UC6.9).

3.9.8 UC6.8: Errore nella modifica di un record

Attori: Amministratore.
Scopo e descrizione: L'attore vuole modificare un record.
Precondizioni: L'attore sta visualzzando il *form* di modifica del record.
Scenario principale:

- L'attore inserisce uno o più valori non validi in uno o più dei campi disponibili;
- L'attore visualizza un messaggio di errore, il quale descrive il valore invalido inserito e l'errore riscontrato.

Postcondizioni: Il messaggio di errore rimane in cima al *form* di modifica e la modifica viene prevenuta.

3.9.9 UC6.9: Annullamento operazione

Attori: Amministratore.
Scopo e descrizione: L'attore vuole annullare l'operazione in corso.
Precondizioni: L'attore non ha ancora terminato l'operazione precedentemente selezionata.
Scenario principale: L'attore clicca sul pulsante "Annulla".
Postcondizioni: L'attore ha annullato correttamente l'operazione ed eventuali modifiche vengono prevenute.

3.9.10 UC6.10: Gestione di un utente

Attori: Amministratore.
Scopo e descrizione: L'attore vuole gestire un utente autorizzato.
Precondizioni: L'attore ha seleziona la scheda "Utenti".
Scenario principale:

- L'attore visualizza il record corrispondente all'utente desiderato (UC6.5);
- L'attore genera una nuova password per l'utente (UC6.10.1).

Postcondizioni: L'attore ha eseguito tutte le operazioni di gestione dell'utente desiderato con successo.

3.9.11 UC6.10.1: Generazione nuova password per un utente

Attori: Amministratore.
Scopo e descrizione: L'attore vuole generare una nuova password per un utente autorizzato.
Precondizioni: L'attore sta visualizzando il record relativo all'utente.
Scenario principale: L'attore preme sul pulsante "Rigenera password".
Postcondizioni: L'attore ha correttamente generato una nuova password per l'utente desiderato.

3.9.12 UC6.11: Gestione di un cliente

Attori: Amministratore.
Scopo e descrizione: L'attore vuole gestire un cliente.

Precondizioni: L'attore ha seleziona la scheda "Clienti".
Scenario principale:

- L'attore visualizza il record corrispondente all'utente desiderato (UC6.5);
- L'attore scarica il modulo di consenso al trattamento dei dati (UC6.11.1).

Postcondizioni: L'attore ha eseguito tutte le operazioni di gestione del cliente desiderato con successo.

3.9.13 UC6.11.1: Scaricamento modulo di consenso al trattamento dei dati

Attori: Amministratore.
Scopo e descrizione: L'attore vuole scaricare il modulo per il consenso al trattamento dei dati firmato da un cliente (se presente).
Precondizioni: L'attore sta visualizzando il record relativo al cliente desiderato.
Scenario principale: L'attore preme sul pulsante "Scarica privacy policy". **Postcondizioni:** L'attore ha correttamente scaricato il modulo (se presente).

3.10 UC7: Visualizzazione dashboard

Attori: Amministratore.
Scopo e descrizione: L'attore vuole visualizzare la *dashboard* del pannello di amministrazione.
Precondizioni: L'attore sta visualizzando una pagina diversa dalla *dashboard*.
Scenario principale: L'attore seleziona la scheda "Dashboard".
Postcondizioni: L'attore viene reindirizzato alla pagina della *dashboard*.

3.11 UC8: Gestione profilo

Attori: Amministratore.
Scopo e descrizione: L'attore vuole gestire il proprio profilo utente.
Precondizioni: L'attore ha accesso al pannello di amministrazione.
Scenario principale:

- L'attore clicca sull'icona del profilo.
- L'attore visualizza il record relativo al proprio profilo (UC6.5)

Postcondizioni: L'attore ha gestito correttamente il proprio profilo.

3.12 UC9: Disconnessione

Attori: Amministratore.
Scopo e descrizione: L'attore vuole disconnettersi dal pannello di amministrazione.
Precondizioni: L'attore ha accesso al pannello di amministrazione.
Scenario principale: L'attore clicca sull'icona di disconnessione.
Postcondizioni: L'attore viene correttamente disconnesso dal pannello di amministrazione.

3.13 UC G2: Operazioni per utente farmacista

Questo caso d'uso generale riassume le operazioni disponibili al *front end* quando è attiva una sessione per un utente con ruolo di farmacista. Come anticipato, le operazioni vengono considerate non dal punto di vista dell'utente finale della piattaforma, cioè il farmacista o cliente che si interfaccia con il *front end*, ma come richieste che l'istanza di *front end* può effettuare alle API del servizio *back end*. Non verranno, per questo motivo, quindi fatte distinzioni tra interazioni che verrebbero effettuate da un farmacista o da un cliente. Infine, va considerato implicito che ognuna delle seguenti operazioni è ristretta ai record del database resi visibili dalla sessione attiva (nella fattispecie i record associati alla farmacia a cui appartiene l'utente autenticato).
Attori: Front end.
Scopo e descrizione: L'attore vuole eseguire operazioni permesse ad un utente con ruolo di farmacista.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale:

- L'attore modifica i dati della farmacia (UC10);
- L'attore gestisce gli utenti autorizzati (UC11);

- L'attore gestisce gli esami (UC12);
- L'attore gestisce i clienti (UC13);
- L'attore gestisce le spedizioni (UC14);
- L'attore effettua la disconnessione dalla piattaforma (UC15).

Postcondizioni: L'attore ha correttamente portato a termine le operazioni desiderate.

3.14 UC10: Aggiornamento dati della farmacia

Attori: Front end.
Scopo e descrizione: L'attore vuole aggiornare i dati della farmacia corrente.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale: L'attore invia una richiesta di aggiornamento del record della farmacia con i nuovi dati, potenzialmente allegando un'immagine da sostituire al logo corrente.
Postcondizioni: Il record corrispondente alla farmacia viene aggiornato e l'attore riceve una risposta che conferma il successo dell'operazione.

3.15 UC11: Gestione utenti autorizzati

Attori: Front end.
Scopo e descrizione: L'attore vuole gestire gli altri utenti autorizzati appartenenti alla farmacia corrente.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale:

- L'attore visualizza la lista degli utenti autorizzati;
- L'attore visualizza un utente autorizzato;
- L'attore richiede il reset della password di un utente autorizzato;
- L'attore imposta una nuova password per un utente autorizzato;
- L'attore crea un utente autorizzato;

- L'attore elimina un utente autorizzato.

Postcondizioni: L'attore ha portato a termine con successo le operazioni di gestione degli utenti.

3.15.1 UC11.1: Visualizzazione lista degli utenti autorizzati

Attori: Front end.
Scopo e descrizione: L'attore vuole visualizzare la lista degli utenti autorizzati per la farmacia corrente.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale: L'attore richiede al sistema i dati degli utenti autorizzati appartenenti alla farmacia corrente. **Postcondizioni:** L'attore riceve correttamente i dati richiesti dal sistema. **Estensioni:** L'attore invia una richiesta non valida (UC16).

3.15.2 UC11.2: Visualizzazione utente autorizzato

Attori: Front end.
Scopo e descrizione: L'attore vuole visualizzare un utente autorizzate per la farmacia corrente.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale: L'attore richiede al sistema i dati di un utente autorizzato appartenente alla farmacia corrente. **Postcondizioni:** L'attore riceve correttamente i dati richiesti dal sistema. **Estensioni:** L'attore invia una richiesta non valida (UC16).

3.15.3 UC11.3: Richiesta reset della password di un utente autorizzato

Attori: Front end.
Scopo e descrizione: L'attore vuole richiedere il reset della password per un utente autorizzato.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente

con ruolo di farmacista.

Scenario principale:

- L'attore richiede al sistema il reset della password di un utente autorizzato appartenente alla farmacia corrente.
- Il sistema invia un'email per la procedura di reset all'indirizzo mail dell'utente specificato.

Postcondizioni: La richiesta di reset viene processata correttamente dal sistema. **Estensioni:** L'attore invia una richiesta non valida (UC16).

3.16 UC11.4: Impostazione nuova password per un utente autorizzato

Attori: Front end.
Scopo e descrizione: L'attore vuole impostare.
Precondizioni: L'attore è collegato al sistema, possiede una sessione attiva per un utente con ruolo di farmacista e possiede un *token* valido per la reimpostazione della password per l'utente autorizzato desiderato.
Scenario principale: L'attore invia al sistema la richiesta di reimpostazione contenente il *token* di reset e la nuova password desiderata. **Postcondizioni:** Il sistema reimposta correttamente la password per l'utente desiderato e l'attore riceve un messaggio di conferma del successo dell'operazione. **Estensioni:** L'attore invia una richiesta non valida (UC16).

3.16.1 UC11.5: Creazione utente autorizzato

Attori: Front end.
Scopo e descrizione: L'attore vuole visualizzare un utente autorizzate per la farmacia corrente.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale: L'attore richiede al sistema i dati di un utente autorizzato appartenente alla farmacia corrente. **Postcondizioni:** L'attore riceve correttamente i dati richiesti dal sistema. **Estensioni:** L'attore invia una richiesta non valida (UC16).

3.17 UC13: Gestione clienti

Attori: Front end.
Scopo e descrizione: L'attore vuole gestire i clienti appartenenti alla farmacia corrente.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale:

- L'attore visualizza il registro dei clienti;
- L'attore crea un cliente;
- L'attore elimina un cliente;
- L'attore modifica i dati di un cliente;

Postcondizioni: L'attore ha portato a termine con successo le operazioni di gestione dei clienti.

3.18 UC14: Gestione spedizioni

Attori: Front end.
Scopo e descrizione: L'attore vuole gestire le spedizioni per la farmacia corrente.
Precondizioni: L'attore è collegato al sistema e possiede una sessione attiva per un utente con ruolo di farmacista.
Scenario principale:

- L'attore visualizza la lista delle spedizioni;
- L'attore visualizza una spedizione;
- L'attore crea una spedizione;
- L'attore elimina una spedizione;
- L'attore modifica i dati di una spedizione;
- L'attore visualizza la lettera di vettura per una spedizione;
- L'attore prenota il ritiro di una spedizione.

Postcondizioni: L'attore ha portato a termine con successo le operazioni di gestione dei clienti.

3.19 UC16: Invio di una richiesta invalida

Attori: Front end.
Scopo e descrizione: L'attore vuole inviare una richiesta al sistema.
Precondizioni: L'attore è collegato al sistema e possiede una richiesta contenente dati non validi.
Scenario principale:

- L'attore invia la richiesta con dati non validi;
- L'attore riceve un messaggio di errore che descrive la violazione commessa.

Postcondizioni: La richiesta dell'attore viene rifiutata e l'attore ha ricevuto un messaggio di errore appropriato.

Nulla facilisi. In vel sem. Morbi id urna in diam dignissim feugiat. Proin molestie tortor eu velit. Aliquam erat volutpat. Nullam ultrices, diam tempus vulputate egestas, eros pede varius leo.

Quoteauthor Lastname

4

Progettazione

5
Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetuer erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis. Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus. Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis.

Cras sed ante. Phasellus in massa. Curabitur dolor eros, gravida et, hendrerit ac, cursus non, massa. Aliquam lorem. In hac habitasse platea dictumst. Cras eu mauris. Quisque lacus. Donec ipsum. Nullam vitae sem at nunc pharetra ultricies. Vivamus elit eros, ullamcorper a, adipiscing sit amet, porttitor ut, nibh. Maecenas adipiscing mollis massa. Nunc ut dui eget nulla venenatis aliquet. Sed luctus posuere justo. Cras vehicula varius turpis. Vivamus eros metus, tristique sit amet, molestie dignissim, malesuada et, urna.

Cras dictum. Maecenas ut turpis. In vitae erat ac orci dignissim eleifend. Nunc quis justo. Sed vel ipsum in purus tincidunt pharetra. Sed pulvinar, felis id consectetuer malesuada, enim nisl mattis elit, a facilisis tortor nibh quis leo. Sed augue lacus, pretium vitae, molestie eget, rhoncus quis, elit. Donec in augue. Fusce orci wisi, ornare id, mollis vel, lacinia vel, massa.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

Quisque facilisis erat a dui. Nam malesuada ornare dolor. Cras gravida, diam sit amet rhoncus ornare, erat elit consectetuer erat, id egestas pede nibh eget odio. Proin tincidunt, velit vel porta elementum, magna diam molestie sapien, non aliquet massa pede eu diam. Aliquam iaculis. Fusce et ipsum et nulla tristique facilisis. Donec eget sem sit amet ligula viverra gravida. Etiam vehicula urna vel turpis. Suspendisse sagittis ante a urna. Morbi a est quis orci consequat rutrum. Nullam egestas feugiat felis. Integer adipiscing semper ligula. Nunc molestie, nisl sit amet cursus convallis, sapien lectus pretium metus, vitae pretium enim wisi id lectus. Donec vestibulum. Etiam vel nibh. Nulla facilisi. Mauris pharetra. Donec augue. Fusce ultrices, neque id dignissim ultrices, tellus mauris dictum elit, vel lacinia enim metus eu nunc.

Ringraziamenti

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.