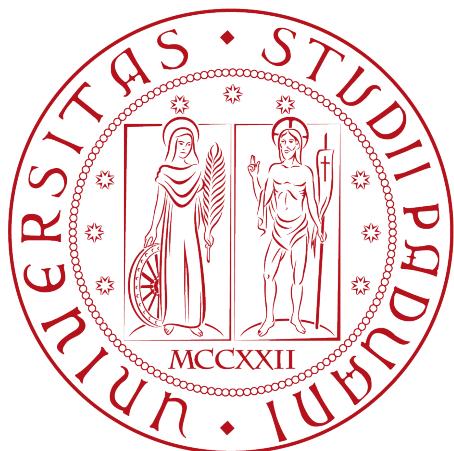


Università degli Studi di Padova
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"
CORSO DI LAUREA IN INFORMATICA



**Single Page Application: tecnologie
innovative per il front-end e per il back-end**

Tesi di laurea triennale

Relatore

Prof. Francesco Ranzato

Laureando

Claudio Zanacchi

ANNO ACCADEMICO 2016-2017

Claudio Zanacchi: *Single Page Application: tecnologie innovative per il front-end e per il back-end*, Tesi di laurea triennale, © Dicembre 2017.

Il mio fine è fare di ogni fine un nuovo inizio.

— Caparezza

Sommario

Questa tesi di laurea è una relazione finale sulla mia attività di stage esterno svolta presso l’azienda IKS s.r.l. di Padova nel periodo che va dal 19 luglio al 29 settembre 2017, per un totale di 312 ore di lavoro. Nel seguente documento verranno descritte in dettaglio l’analisi dei requisiti, la progettazione, l’implementazione e la verifica di alcune parti dell’applicazione web *Logistic Performance*, soluzione software di IKS nell’ambito della logistica, oltre che le tecnologie innovative (sia lato front-end che lato back-end) che sono state utilizzate durante tutto il periodo di stage.

Tutti i diagrammi delle classi, dei package e dei casi d’uso (presenti nei Capitoli 3 e 4) sono conformi allo standard *UML 2.0*. Per realizzarli è stato usato il software *Astah Professional*.

L’intero lavoro è stato svolto in ambiente *Linux Ubuntu 14.4 LTS* ad eccezione delle parti riguardanti la realizzazione dei mockup: questo lavoro è stato svolto in ambiente *Microsoft Windows 10* in quanto il software usato per realizzarli, *Adobe Experience Design* non è disponibile per la piattaforma *Linux*.

Capitolo 1 descrive l’azienda e il progetto di stage assegnato.

Capitolo 2 fornisce una carrellata di brevi descrizioni delle tecnologie utilizzate durante tutto il periodo di stage.

Capitolo 3 formalizza tutti i casi d’uso e i requisiti ad alto livello raccolti in fase di analisi dei requisiti.

Capitolo 4 illustra in modo esaustivo la progettazione del prodotto a diversi livelli di astrazione.

Capitolo 5 descrive la fase di verifica e di validazione delle funzionalità del prodotto finale a seguito della fase di implementazione.

Capitolo 6 fornisce una panoramica generale sia riguardo alle risorse utilizzate, sia riguardo alle impressioni personali sullo stage dopo la conclusione dello stesso.

In fondo al documento si trovano tre appendici.

Appendice A contiene tutti i verbali volti a riassumere il contenuto delle principali riunioni svolte in azienda con il Tutor Aziendale.

Appendice B illustra in sintesi la storia e le funzionalità innovative del framework Angular.

Appendice C parla degli aspetti che hanno portato all'enorme successo della piattaforma cloud AWS.

Per quanto riguarda la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- * Gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento. Per la prima occorrenza dei termini riportati nel glossario viene utilizzata una nomenclatura con un lettera G a pedice in questo modo: *parola_G*;
- * I termini in lingua straniera meno frequenti o facenti parte del gergo tecnico sono evidenziati con il carattere *corsivo*.
- * I termini che identificano nomi di classi, metodi e componenti e in generale qualsiasi parte di codice sorgente sono evidenziati con il font monospaziato **font**.
- * I riferimenti bibliografici vengono indicati con un apice, in questo modo: *parola^[numero fonte]*.

Indice

1	Introduzione	1
1.1	Dominio aziendale	1
1.1.1	Azienda ospitante	1
1.1.2	Ambiti aziendali	2
1.2	Progetto aziendale	2
1.2.1	Introduzione alle Single Page Application	2
1.2.2	Introduzione al progetto	3
1.2.3	Descrizione del progetto	4
1.3	Pianificazione	8
1.3.1	Fase 1	8
1.3.2	Fase 2	8
1.3.3	Fase 3	9
1.3.4	Fase 4	9
1.3.5	Fase 5 (Opzionale)	9
2	Tecnologie utilizzate	11
2.1	Tecnologie lato Front-end	11
2.1.1	Angular 4	11
2.1.2	TypeScript	12
2.1.3	HTML	12
2.1.4	CSS	13
2.2	Tecnologie lato Back-end	13
2.2.1	AWS Lambda	13
2.2.2	Amazon API Gateway	14
2.2.3	Amazon DynamoDB	15
2.2.4	Amazon S3	15
2.2.5	Amazon Cognito	15
2.3	Tecnologie di supporto	17
2.3.1	Visual Studio Code	17
2.3.2	Git	17
2.3.3	AWS CodeCommit	18
2.3.4	AWS IAM	18
2.3.5	Adobe Experience Design	18
2.3.6	Tema grafico Inspinia	19
2.3.7	Swagger	19
2.3.8	Slack	20
3	Analisi dei requisiti	21

3.1	Pagina di dettaglio di una consegna	22
3.1.1	UC1: Visualizzazione e modifica della lista di proprietà di una consegna	22
3.1.2	UC2: Visualizzazione della localizzazione	23
3.1.3	UC3: Visualizzazione bolla e orario della consegna	24
3.1.4	UC4: Navigazione	25
3.2	Pagina della lista degli utenti	25
3.2.1	UC5: Visualizzazione della lista degli utenti	26
3.3	Pagina di dettaglio di un singolo utente	27
3.3.1	UC6: Visualizzazione e modifica della lista di proprietà di un utente	27
3.4	Pagina di creazione di un nuovo utente	28
3.4.1	UC7: Creazione di un nuovo utente	28
3.5	Requisiti	29
4	Progettazione e sviluppo	33
4.1	Prototipizzazione dell'interfaccia utente	33
4.1.1	Pagina di dettaglio di una consegna	34
4.1.2	Pagina della lista degli utenti	36
4.1.3	Pagina di dettaglio di un singolo utente	39
4.1.4	Pagina di creazione di un nuovo utente	41
4.2	Progettazione architettonale	43
4.3	Progettazione di dettaglio e codifica	54
5	Test e Validazione	73
5.1	Modalità di esecuzione dei test	73
5.2	Test	73
6	Valutazione retrospettiva	77
6.1	Raggiungimento degli obiettivi prefissati	77
6.2	Consuntivo orario finale	78
6.3	Problematiche incontrate	79
6.4	Conoscenze acquisite	79
6.5	Valutazione personale sullo stage	80
6.6	Valutazione personale sul corso di studi	81
A	Verbali	83
B	L'evoluzione del framework Angular	89
C	Il successo di AWS	93
	Bibliografia	97
	Glossario	99

Elenco delle figure

1.1	Logo di IKS	1
1.2	Vista 1 dell'applicazione	5
1.3	Vista 2 dell'applicazione	6
1.4	Diagramma di <i>Gantt</i> della pianificazione dello stage	8
2.1	Logo di Angular 4	12
2.2	Logo di TypeScript	12
2.3	Logo di HTML5	13
2.4	Logo di CSS3	13
2.5	Logo di AWS Lambda	14
2.6	Logo di Amazon API Gateway	14
2.7	Logo di Amazon DynamoDB	15
2.8	Logo di Amazon S3	15
2.9	Logo di Amazon Cognito	16
2.10	Logo di Visual Studio Code	17
2.11	Logo di Git	17
2.12	Logo di AWS CodeCommit	18
2.13	Logo di AWS IAM	18
2.14	Logo di Adobe Experience Design	19
2.15	Interfaccia del tema grafico Insipinia	19
2.16	Logo di Swagger	20
2.17	Logo di Slack	20
3.1	UC1: Visualizzazione e modifica della lista di proprietà di una consegna	22
3.2	UC2: Visualizzazione della localizzazione	23
3.3	UC3: Visualizzazione bolla e orario della consegna	24
3.4	UC5: Visualizzazione della lista degli utenti	26
3.5	UC6: Visualizzazione e modifica della lista di proprietà di un utente	27
3.6	UC7: Creazione di un nuovo utente	29
4.1	Wireframe di <i>Pagina di dettaglio di una consegna</i>	34
4.2	Wireframe con dettaglio delle aree di <i>Pagina di dettaglio di una consegna</i>	35
4.3	Mockup di <i>Pagina di dettaglio di una consegna</i>	35
4.4	Mockup della User Interface di <i>Pagina di dettaglio di una consegna</i>	36
4.5	Wireframe di <i>Pagina della lista degli utenti</i>	36
4.6	Wireframe con dettaglio delle aree di <i>Pagina della lista degli utenti</i>	37
4.7	Mockup di <i>Pagina della lista degli utenti</i>	38
4.8	Mockup della User Interface di <i>Pagina della lista degli utenti</i>	38

4.9	Wireframe di <i>Pagina di dettaglio di un singolo utente</i>	39
4.10	Wireframe con dettaglio delle aree di <i>Pagina di dettaglio di un singolo utente</i>	39
4.11	Mockup di <i>Pagina di dettaglio di un singolo utente</i>	40
4.12	Mockup della User Interface di <i>Pagina di dettaglio di un singolo utente</i>	41
4.13	Wireframe di <i>Pagina di creazione di un nuovo utente</i>	41
4.14	Wireframe con dettaglio delle aree di <i>Pagina di creazione di un nuovo utente</i>	42
4.15	Mockup di <i>Pagina di creazione di un nuovo utente</i>	42
4.16	Mockup della User Interface di <i>Pagina di creazione di un nuovo utente</i>	43
4.17	app	44
4.18	delivery	45
4.19	users	46
4.20	appviews	47
4.21	users-table	48
4.22	user-detail	49
4.23	create-user	50
4.24	delivery-detail	50
4.25	property-list-detail	51
4.26	geolocation-detail	52
4.27	photo-detail	53
4.28	DeliveryDetailComponent	54
4.29	TransformerDelivery	55
4.30	PropertyListDetailComponent	55
4.31	PropertyRow	56
4.32	GeolocationDetailComponent	57
4.33	GeolocationProperty	58
4.34	PhotoDetailComponent	58
4.35	PhotoProperty	59
4.36	Bucket	59
4.37	Destination	60
4.38	Localization	61
4.39	Delivery	61
4.40	DeliveryService	63
4.41	UsersTableComponent	63
4.42	AttributeProperty	65
4.43	User	65
4.44	UserTable	66
4.45	UserService	67
4.46	UserPropertyRow	68
4.47	UserDetailComponent	69
4.48	CreateUserComponent	70
5.1	Console AWS	73
5.2	Esempio 1 di test per il design responsive	74
5.3	Esempio 2 di test per il design responsive	74
B.1	Logo di Angular	89
B.2	Confronto tra sintassi TypeScript e sintassi JavaScript di codice equivalente	91

C.1	Logo di AWS	93
C.2	Infrastruttura globale di AWS	94

Elenco delle tabelle

3.1	Tabella dei Requisti Funzionali	30
3.2	Tabella dei Requisti Non Funzionali	32
5.1	Tabella dei Test	76

Capitolo 1

Introduzione

In questo capitolo viene brevemente descritta l'azienda ospitante in cui è stata svolta l'attività di stage.

Viene inoltre descritto ad alto livello il progetto *Logistic Performance* e vengono illustrate in dettaglio le fasi del piano di lavoro originalmente assegnato comprensive di durata prevista in ore lavorative.

1.1 Dominio aziendale

1.1.1 Azienda ospitante

L'azienda IKS (Information Knowledge Supply) s.r.l. nasce alla fine degli anni novanta con lo scopo di occuparsi dei principali temi dell'Information e Communication Techonology (ICT). Si focalizza oggi su diverse aree strategiche: sviluppo, sicurezza informatica, implementazione, automazione, architetture complesse e innovative e IT governance.

Inoltre è leader di mercato sui temi dell'antifrode in ambito bancario con soluzioni ad hoc e ad alto rendimento. Le soluzioni software sviluppate da IKS sono spesso basate sulla tecnologia Java, ma l'azienda progetta e realizza architetture aperte ai nuovi paradigmi di sviluppo orientati al web e al mobile. IKS ha anche sviluppato negli anni un'approfondita conoscenza delle soluzioni open source, ed è inoltre in grado di sfruttare i nuovi paradigmi *cloud-native*, architetture e microservizi *serverless* e sviluppo applicativo in cui si possa rispondere per tempi e costi alle esigenze del cliente.

Il gruppo ha sedi a Padova, Milano e Trento ed è oggi il partner di riferimento per le molte aziende che necessitano di garantire l'operatività di servizi *business critical*_G.



Figura 1.1: Logo di IKS

1.1.2 Ambiti aziendali

Vengono di seguito citate alcune aree di interesse (accompagnate da una breve descrizione) in cui IKS si cimenta per offrire soluzioni di successo ai suoi clienti.^[1]

Sicurezza

- * **Information Security Management:** consulenza e soluzioni per supportare le aziende nella definizione dei processi e nella predisposizione degli strumenti necessari per il “Sistema di Gestione della Sicurezza delle Informazioni”;
- * **Fraud Management:** ambito di intervento mirato al contrasto di operazioni fraudolente in ambito bancario;
- * **Gestione identità e accessi:** soluzioni per la gestione del ciclo di vita delle identità nel sistema informativo con funzionalità di *User Administration* e *Provisioning*;
- * **Sicurezza dei dati:** soluzioni per la protezione dei dati;
- * **Sicurezza delle applicazioni:** soluzioni per la protezione delle applicazioni tramite tecnologie IPS (Intrusion Prevention System) e WAF (Web Application Firewall) in grado di proteggere le applicazioni da utilizzi impropri mirati al furto di informazioni;
- * **Sicurezza delle reti e dei contenuti:** soluzioni per la difesa del perimetro aziendale, sia esso fisico, virtuale o Cloud-based.

Architetture applicative

- * **Basi di dati:** gestione di basi di dati di diverse tipologie. Le informazioni memorizzate possono essere non strutturate e non relazionate. Esse vengono memorizzate direttamente nei formati in cui vengono utilizzate;
- * **Back-end:** architetture e soluzioni software in grado di far accedere in modalità gestita e controllata le informazioni interne alla propria azienda direttamente dal web.
- * **Front-end:** soluzioni software per la scrittura di applicazioni che si basano sul paradigma delle chiamate a API;
- * **API management:** soluzioni software che si occupano di tutti quegli aspetti che garantiscono la sicurezza, il monitoraggio, lo sviluppo, il versionamento e le prestazioni delle transazioni in ingresso e in uscita e di disaccoppiare l’infrastruttura interna dal mondo web.

1.2 Progetto aziendale

1.2.1 Introduzione alle Single Page Application

Tradizionalmente, un sito web non è altro che un insieme di pagine web (ovvero documenti digitali tipicamente scritti in linguaggio di markup HTML) interconnesse tra loro (tramite collegamenti ipertestuali) che risiedono su uno o più server remoti. Durante la navigazione di un utente, il browser (programma che risiede sulla macchina

dell’utente, in grado di elaborare le informazioni contenute nelle pagine web) non fa altro che scaricare dal server ogni singola pagina web di cui l’utente ha bisogno e visualizzarla staticamente; in questo modo per visualizzare eventuali aggiornamenti dei contenuti della pagina web, l’utente sarà costretto a ricaricare la pagina (il browser la rimpiazzerà interamente con una sua versione più aggiornata scaricata dal server).

Al giorno d’oggi, il numero dei siti web completamente statici si sta riducendo sempre di più ed essi stanno pian piano lasciando il posto alle applicazioni web, ovvero veri e propri programmi molto più simili ad applicazioni per desktop che a documenti di testo, che vengono eseguiti direttamente su un browser web. Questo rapido cambiamento del web ha portato numerosi vantaggi sia in termini di usabilità che di manutenibilità, in quanto l’interazione tra il client e il server avviene in modo molto più dinamico.

Una *Single Page Application* (ovvero applicazione su singola pagina) non è altro che un’applicazione web che può essere eseguita su una singola pagina web con l’obiettivo di fornire una esperienza utente di gran lunga più fluida e più simile alle applicazioni desktop dei tradizionali sistemi operativi. Contrariamente a un sito web statico, infatti, una Single Page Application aggiorna continuamente la pagina web corrente in modo totalmente dinamico ad ogni interazione dell’utente piuttosto che caricare intere nuove pagine da un server remoto. I vantaggi di questo approccio sono molteplici: poichè il codice viene eseguito principalmente lato client, al server saranno inviate meno richieste, e questo garantirà un sostanzioso aumento delle performance. Inoltre, una Single Page Application può integrarsi meglio con qualsiasi framework, in quanto la comunicazione con il back-end avviene tramite un formato standard, ovvero il JSONG.

Front-end

La parte di front-end di una applicazione web non è altro che la parte visibile dell’applicazione, ovvero quella con cui l’utente può interagire direttamente o estrarne le informazioni di proprio interesse. Questa parte è dunque responsabile nell’acquisizione dei dati in ingresso e della loro elaborazione. Chi lavora al front-end di una applicazione solitamente possiede solide conoscenze di *web usability*, in modo da fornire all’utente una esperienza d’utilizzo dell’applicazione semplice ed intuitiva.

Back-end

La parte di back-end di una applicazione web è invece la parte che si occupa del suo funzionamento logico. Tipicamente i risultati richiesti dagli utenti tramite l’interazione con la grafica dell’applicazione (ovvero la parte di front-end) vengono restituiti tramite un sistema di interrogazioni a un qualche database che risiede su un server remoto.

1.2.2 Introduzione al progetto

L’applicazione web *Logistic Performance* nasce come soluzione di IKS per supportare le attività di logistica di uno dei suoi clienti, che da un centro di produzione ha la necessità di distribuire i suoi prodotti a vari centri di distribuzione in diverse parti del mondo. Il cliente si appoggia a più ditte di trasportatori ed è interessato a sapere quali di queste lavorano meglio (ovvero quali sono le più puntuali, o quali consegnino la merce nello stato migliore). Questa applicazione offre un supporto alla tracciabilità di queste consegne, oltre che un sistema di verifica sia delle performance dei trasportatori

che dello stato di questi viaggi.

IKS ha già implementato una soluzione complessa di questa applicazione. Questa soluzione presenta un back-end sviluppato in linguaggio di programmazione PHP e un flusso di integrazione con sistemi informativi del cliente su cloud. Ha una versione desktop e una versione mobile (quest'ultima è supportata sia dal sistema Android che dal sistema iOS). Nella versione mobile i trasportatori possono fotografare la bolla di consegna con la firma del destinatario direttamente utilizzando la fotocamera del dispositivo e caricarla immediatamente con tanto di posizione GPS in modo che il cliente possa ottenere un feedback immediato sullo stato del viaggio. I problemi con cui il cliente si deve confrontare sono molteplici: una consegna potrebbe arrivare in ritardo oppure essere contestata dal ricevente (per esempio perché la merce è in stato di deterioramento, o magari perché mancano alcuni colli). È anche previsto un sistema di autenticazione *enterprise* per la versione mobile: dopo la prima autenticazione il dispositivo resta configurato; in questo modo il trasportatore non sarà costretto ad effettuare l'autenticazione ogniqualvolta si necessiti di una foto di una bolla di consegna.

Il progetto dello stage riguarda il rifacimento da zero di questa applicazione web su tecnologie del tutto nuove sia lato front-end che lato back-end preservandone le funzionalità. In particolare lo scopo di questo rifacimento è il voler migliorare:

- * L'esperienza visiva dell'utente grazie a una grafica più accattivante;
- * L'usabilità dell'applicazione;
- * Le prestazioni dell'applicazione;
- * Le modalità di gestione dell'autenticazione degli utenti.

1.2.3 Descrizione del progetto

Vengono qui presentate due viste ad alto livello della struttura del funzionamento della nuova versione di Logistic Performance. Esse si appoggiano su diversi servizi **AWS (Amazon Web Services)**^[2] per quanto riguarda la parte di back-end, mentre il front-end è interamente realizzato con la tecnologia **Angular 4**^[3].

- La prima vista (Figura 1.2) è relativa al funzionamento di tutte le componenti del sistema la cui parte del front-end interagisce con il back-end solo ed esclusivamente tramite Amazon API Gateway (servizio di AWS), il quale si occupa di eseguire chiamate a funzioni AWS Lambda (altro servizio di AWS). Questa vista è stata il riferimento principale per lo sviluppo della pagina web *Pagina di dettaglio di una consegna*.

- La seconda vista (Figura 1.3) è invece relativa al funzionamento di tutte le componenti del sistema riguardanti la gestione dell'autenticazione, gestita tramite Amazon Cognito (altro servizio di AWS). La parte del front-end interagisce in questo caso con il back-end non più tramite Amazon API Gateway, ma direttamente con metodi asincroni scritti nei sorgenti del front-end in linguaggio JavaScript grazie a un SKD_G JavaScript per il supporto ad Amazon Cognito. Questa vista è stata il riferimento principale per lo sviluppo delle pagine web *Pagina della lista degli utenti*, *Pagina di creazione di un nuovo utente* e *Pagina di dettaglio di un singolo utente*.

Questo paragrafo si concentra soltanto sugli aspetti funzionali del sistema. Tutte le tecnologie di riferimento qui citate saranno descritte più in dettaglio nel Capitolo 2.

Schema dell'architettura

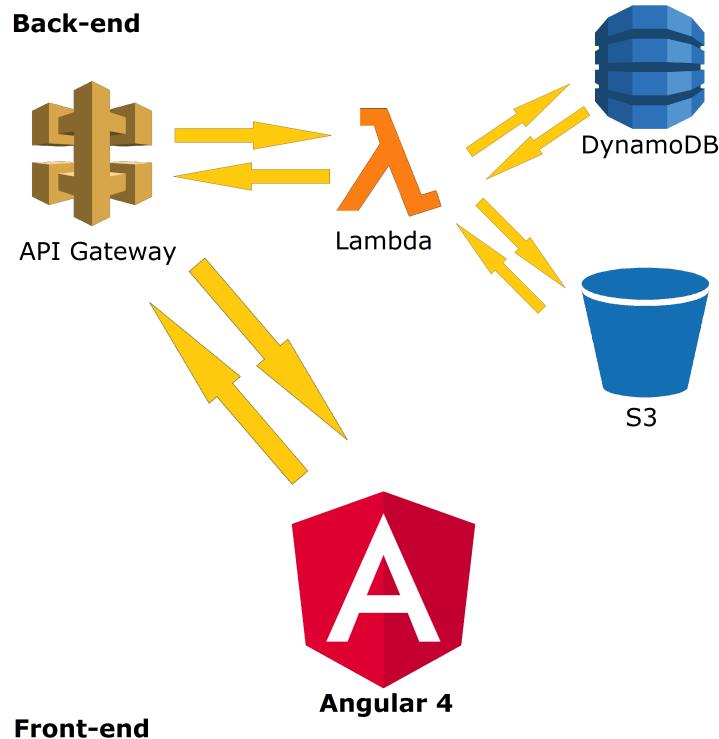


Figura 1.2: Vista 1 dell'applicazione

- * Il cuore del back-end si basa su diverse funzioni **AWS Lambda** (scritte interamente in linguaggio di programmazione Java). AWS Lambda è una piattaforma di calcolo *serverless* che esegue codice in risposta a determinati eventi e gestisce in modo del tutto automatico le risorse richieste dal codice sorgente. Le funzioni Lambda interagiscono con tutte le altre componenti AWS del back-end.
- * Chi controlla gli accessi dall'esterno è **Amazon API Gateway**. Esso è un servizio che consente di creare API che agiscano come porta d'entrata attraverso la quale le applicazioni possono accedere a dati, logica di business o funzionalità dei servizi di back-end. Serve per non esporre direttamente la logica del back-end. È questa componente che si interfaccia con gli eventi generati dal front-end.
- * La persistenza dei dati strutturati dell'applicazione viene fatta servendosi di **Amazon DynamoDB**, database NoSQLG proprio di AWS.
- * La persistenza dei dati non strutturati (come le immagini) viene fatta servendosi di **Amazon S3**, che è uno storageG di oggetti creato per memorizzare e ripristinare un volume di dati qualsiasi da ogni possibile origine. Anche la parte di front-end dell'applicazione è al momento deployata su Amazon S3.

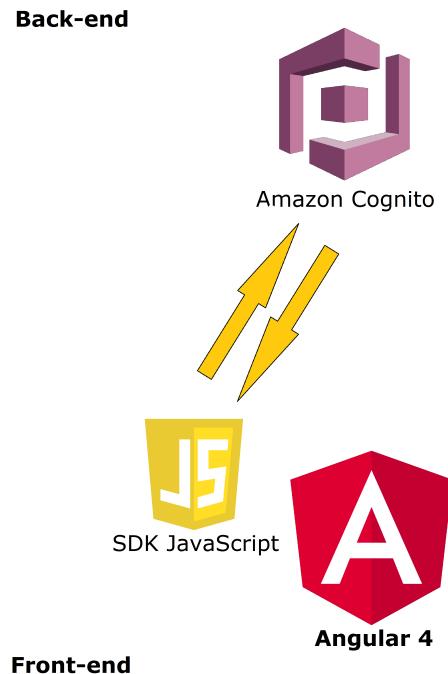


Figura 1.3: Vista 2 dell'applicazione

- * Il servizio che si occupa della gestione e dell'autenticazione degli utenti e della sincronizzazione su più dispositivi è **Amazon Cognito**. Il front-end dell'applicazione interagisce con Amazon Cognito tramite un SDK JavaScript, che aggiunge una vasta gamma di funzionalità (sotto forma di metodi scritti in JavaScript) molto utili. Grazie a queste nuove funzionalità è possibile per esempio creare nuovi utenti con diversi permessi, recuperare la lista di tutti gli utenti dell'applicazione ciascuno con le loro proprietà e altro ancora.
- * Il front-end dell'applicazione è sviluppato in **Angular 4**, framework web open source sviluppato da Google nato con lo scopo di facilitare proprio lo sviluppo di Single Page Application. Tale framework fa abbondantemente uso delle tecnologie HTML, CSS e TypeScript, linguaggio di programmazione sviluppato da Microsoft e superset di JavaScript.

Obiettivi dello stage

Gli obiettivi principali dello stage sono stati definiti e concordati con il Tutor Aziendale. Essi sono i seguenti:

- * Una buona comprensione del funzionamento di tutti gli strumenti necessari per lo sviluppo delle parti dell'applicazione assegnate (principalmente la tecnologia Angular 4);
- * La capacità di adottare un approccio lavorativo mirato alla realizzazione di piccoli ma completi cicli di sviluppo, componente dopo componente, piuttosto che la

realizzazione di un unico grande ciclo di sviluppo con una lavorazione in parallelo su tutte le diverse componenti assegnate;

- * La netta divisione in fasi del ciclo di vita, identificate in:

- Analisi dei requisiti
- Prototipizzazione dell’interfaccia utente (realizzazione di mockup)
- Progettazione architetturale
- Progettazione di dettaglio
- Implementazione
- Verifica delle funzionalità

- * La capacità di coordinamento per lavorare in gruppo con altri sviluppatori, incaricati di realizzare altre componenti della medesima applicazione;

- * Il versionamento dei mockup e di tutto il codice sorgente su un repository Git, tramite la tecnologia **AWS CodeCommit**;

- * La capacità di prendere decisioni con spirito di iniziativa e inventiva singolarmente, muovendosi in autonomia il più possibile.

Scopo del progetto

Lo scopo principale dello stage è stato quello di lavorare su alcune parti dell’applicazione Logistic Performance. Inizialmente era stato deciso di realizzare le seguenti tre parti:

- * **Pagina di dettaglio di una consegna**
- * **Pagina di amministrazione per l’associazione di un nuovo device**
- * **Dashboard di monitoraggio con indicatori riassuntivi**

In seguito, per motivi funzionali e di tempistiche, è stato necessario modificare un buon numero dei requisiti della seconda parte assegnata: la *Pagina di amministrazione per l’associazione di un nuovo device* che inizialmente doveva essere un’unica pagina comprendente sia la gestione dell’autenticazione dell’utente che il riconoscimento di un nuovo device tramite QR-CODE_G non è mai stata realizzata e al suo posto sono invece state realizzate le tre pagine distinte *Pagina della lista degli utenti*, *Pagina di creazione di un nuovo utente* e *Pagina di dettaglio di un singolo utente*, il cui scopo è rimasto comunque quello della gestione dell’autenticazione degli utenti dell’applicazione. Invece la parte relativa alla *Dashboard di monitoraggio con indicatori riassuntivi* è stata del tutto eliminata dal piano di stage. Le componenti che sono state realizzate infine sono state:

- * **Pagina di dettaglio di una consegna**
- * **Pagina della lista degli utenti**
- * **Pagina di creazione di un nuovo utente**
- * **Pagina di dettaglio di un singolo utente**

Spesso in fase di analisi di queste componenti si è deciso di dare priorità diverse ai requisiti individuati per la realizzazione di queste componenti. Per una spiegazione più esaustiva si rimanda al Capitolo 3.

1.3 Pianificazione

È stato necessario redigere un piano di lavoro congiuntamente con il Tutor Aziendale, con lo scopo di fissare tempi (in ore lavorative) e gli obiettivi dello stage organizzati in fasi distinte. Di seguito viene riportata una rappresentazione del piano di lavoro originale tramite un *diagramma di Gantt*.

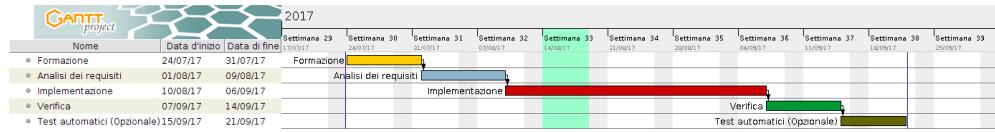


Figura 1.4: Diagramma di Gantt della pianificazione dello stage

1.3.1 Fase 1

Scopo: formazione sul contesto, le problematiche e le tecnologie che si incontreranno durante lo stage. In dettaglio:

- * Servizi cloud Amazon Web Services orientati alle soluzioni serverless (AWS Lambda, supporto linguaggio JAVA)
- * Strumenti e librerie di sviluppo per applicazioni Single Page Application (Angular ver.4)
- * Ambiente di sviluppo e strumenti a supporto (Bitbucket-GIT, JIRA, IntelliJ Idea, ...)

Output:

- * Documentazione riassuntiva sugli strumenti individuati e principali punti di forza
- * Predisposizione ambiente di sviluppo per l'applicazione web con le tecnologie di riferimento

Durata: ~ 48 ore

1.3.2 Fase 2

Scopo: analisi delle specifiche funzionali, definizione del piano di realizzazione e progettazione tecnica della soluzione. In dettaglio:

- * Analisi specifiche funzionali
- * Progettazione di dettaglio
- * Redazione Documentazione

Le componenti applicative di cui è prevista la realizzazione sono:

- * Pagina di dettaglio di una consegna
- * Pagina di amministrazione per l'associazione di un nuovo *device*

- * *Dashboard* di monitoraggio con indicatori riassuntivi

Output:

- * Documento di specifica tecnica

Durata: ~ 56 ore**1.3.3 Fase 3**

Scopo: Implementazione della soluzione con integrazione degli strumenti previsti ed eventuali allineamenti con altri gruppi di lavoro.

Output:

- * Codice del software sviluppato su repository versionato
- * Realizzazione di pacchetti di test per verifiche operative

Durata: ~ 120 ore**1.3.4 Fase 4**

Scopo: Test funzionale della soluzione ed eventuali correzioni. Verifica dei requisiti del progetto di stage. Redazione della documentazione tecnica.

Output:

- * Ambiente di test operativo e funzionalmente completo
- * Documentazione completa della soluzione

Durata: ~ 48 ore**1.3.5 Fase 5 (Opzionale)**

Scopo: Sviluppo e integrazione di procedure di test *End-To-End* per le funzionalità dell'applicazione. Esecuzione dei test in modalità automatica nella pipeline di *Continuous Integration*.

Output:

- * Esempi di test su codice versionato
- * Configurazione di procedure di test automatici su piattaforma AWS.

Durata: ~ 40 ore

Capitolo 2

Tecnologie utilizzate

In questo capitolo seguirà un elenco delle tecnologie di riferimento adottate durante tutto lo stage. Ogni tecnologia sarà accompagnata da una breve descrizione generale e anche da una esposizione dell'ambito di utilizzo nel contesto di questo stage.

2.1 Tecnologie lato Front-end

2.1.1 Angular 4

Angular è un framework web opensource sviluppato da Google che permette lo sviluppo di front-end di Single Page Application performanti e reattive sfruttando il pattern architettonale MVC. Oggi tale framework è giunto alla versione 5; la versione 1 era in precedenza conosciuta come AngularJS¹.

Fornisce inoltre un ottimo supporto al *testing* dell'applicazione e a *design patterns* innovativi come la Dependency Injection. L'architettura di Angular favorisce la scrittura di applicazioni modulari. Questo framework permette allo sviluppatore di creare all'interno della pagina web dei *components*, che rappresentano delle sezioni vere e proprie della pagina web visibili sul browser; ogni *component* ha una propria logica strutturale (scritta tramite appositi marcatori HTML), di presentazione (scritta con appositi fogli di stile CSS) e di business (scritta con il linguaggio di programmazione TypeScript). I *components* dell'applicazione possono comunicare tra di loro scambiandosi oggetti, rispondere a determinati eventi dell'utente e anche essere innestati l'uno dentro l'altro in modo da creare una gerarchia.

Un *service* in Angular è invece una classe che implementa funzionalità che sono condivise dai vari elementi di un'applicazione. Tipicamente queste funzionalità si occupano di recuperare ed elaborare dati provenienti dall'esterno del front-end. Altre funzionalità molto importanti di Angular sono il *data binding*, le *reactive forms* e il *routing*.

Nel contesto dello stage Angular 4 è stata sicuramente la tecnologia più utilizzata in assoluto in quanto è proprio questa la tecnologia sulla quale si basa tutto il front-end dell'applicazione.

¹Per sapere di più sulla storia e sulle funzionalità di Angular, si rimanda all'Appendice B del documento.



Figura 2.1: Logo di Angular 4

2.1.2 TypeScript

TypeScript (giunto oggi alla versione 2.6) è un linguaggio di programmazione open source sviluppato da Microsoft che basa le sue caratteristiche sullo standard ECMAScript 6. Si tratta di un super-set del linguaggio JavaScript (che è il principale linguaggio utilizzato per gestire la logica comportamentale delle pagine web) in quanto estende la sua sintassi: ogni programma scritto in JavaScript infatti rispetta implicitamente la sintassi di TypeScript; inoltre un file sorgente TypeScript, se compilato con successo, produce un file di codice sorgente JavaScript equivalente, e questo consente di scrivere qualsiasi applicazione JavaScript in linguaggio TypeScript, in quanto sarà sempre compatibile con i principali browser di riferimento^[4]. La sintassi di TypeScript fornisce un supporto opzionale alla tipizzazione forte (i tipi di dato non sono obbligatori, ma se specificati, la loro correttezza viene controllata a tempo di compilazione) ed è molto più simile a quella dei linguaggi di programmazione orientati agli oggetti come Java e C++, risultando dunque molto più intuitiva rispetto a quella di JavaScript; in questo modo il programmatore è di gran lunga facilitato nello sviluppo di applicazioni web piuttosto corpose, che richiederebbero solitamente la scrittura di codice JavaScript difficilmente intuitivo e poco comprensibile con l'aumentare della complessità della logica di business. Ogni file scritto in linguaggio TypeScript ha estensione *.ts* e la sua compilazione produce un file in equivalente codice sorgente JavaScript con estensione *.js*.

TypeScript è stato usato come linguaggio per la definizione della logica di business del front-end dell'applicazione. Tutte le classi in dettaglio nel paragrafo 4.3 sono state poi implementate utilizzando questo linguaggio di programmazione.



Figura 2.2: Logo di TypeScript

2.1.3 HTML

HTML (acronimo di HyperText Markup Language) è un linguaggio di markup usato solitamente per la formattazione e impaginazione di documenti ipertestuali disponibili nel World Wide Web sotto forma di pagine web. Questo linguaggio descrive le modalità di impaginazione o visualizzazione grafica (*layout*) del contenuto, testuale e non, di

una pagina web attraverso tag di formattazione. Oggi tale linguaggio è giunto alla versione 5 (conosciuta come HTML5).

HTML è stato usato come linguaggio per la definizione della logica strutturale del front-end dell'applicazione.



Figura 2.3: Logo di HTML5

2.1.4 CSS

CSS (acronimo di Cascading Style Sheets) è un linguaggio usato per definire la presentazione (ovvero l'aspetto grafico vero e proprio) oltre che la formattazione di pagine web scritte in HTML. Oggi è giunto alla versione 3 (con il nome di CSS3).

CSS è stato usato solo in parte come linguaggio per la definizione della logica di presentazione del front-end dell'applicazione in quanto avendo scelto un tema già esistente per gestire aspetti legati alla grafica e al *responsive design* il suo utilizzo è stato limitato.



Figura 2.4: Logo di CSS3

2.2 Tecnologie lato Back-end

Tutte le tecnologie del back-end dell'applicazione sono basate su servizi propri di AWS².

2.2.1 AWS Lambda

AWS Lambda è un servizio di elaborazione *serverless* che può eseguire del codice fornito direttamente dall'utente in uno dei linguaggi di programmazione supportati (attualmente sono supportati solo i linguaggi C#, Java 8, Node.js 4.3, Node.js 6.10, Python 2.7 e Python 3.6).

²Per sapere di più sulla storia e sul successo dei servizi AWS, si rimanda all'Appendice C del documento.

Il codice che l’utente decide di caricare su AWS Lambda viene chiamato funzione Lambda. Una volta creata, una funzione Lambda può essere attivata e quindi eseguita in qualsiasi momento al verificarsi di un evento per il quale la funzione è stata progettata. Le funzioni Lambda sono *stateless*, ovvero prive di qualsiasi affinità con l’intera infrastruttura sottostante.

Nel contesto dello stage è stato in un primo momento necessario visionare tutto il codice sorgente delle funzioni Lambda in linguaggio Java 8; queste funzioni erano utilizzate prevalentemente per implementare il DAO (Data Access Object): dopo aver caricato la funzione su AWS Lambda, infatti, era possibile associarla con altre risorse AWS che riguardavano la persistenza di dati (ad esempio un *bucket* Amazon S3 o una tabella Amazon DynamoDB). Al variare della risorsa, o viceversa, al verificarsi di un evento esterno (ad esempio una richiesta HTTP tramite Amazon API Gateway), Lambda eseguiva la funzione e gestiva le risorse di elaborazione in modo da soddisfare le esigenze dettate dalle richieste in entrata.



Figura 2.5: Logo di AWS Lambda

2.2.2 Amazon API Gateway

Amazon API Gateway è un servizio il cui scopo è quello di semplificare agli sviluppatori la creazione, la pubblicazione, la manutenzione, il monitoraggio e la protezione delle API su qualsiasi scala. È possibile creare con questo servizio una API che si comporti come una porta d’ingresso attraverso la quale applicazioni possano accedere a dati, logica di business o funzionalità dei servizi del back-end di una applicazione.

Nel contesto dello stage è servito prevalentemente per il controllo dell’accesso dell’intera logica del back-end dell’applicazione (e quindi a non esporla direttamente) da parte dei servizi del front-end.



Figura 2.6: Logo di Amazon API Gateway

2.2.3 Amazon DynamoDB

Amazon DynamoDB è un servizio di database NoSQL iperperformante pensato per tutte le applicazioni che richiedono una latenza costante non superiore a una decina di millisecondi su qualsiasi scala. È un database *cloud* che supporta sia i modelli di *storage document* sia quelli di tipo chiave-valore. Per utilizzarlo è sufficiente creare una tabella; tutte le attività di gestione del database, come il *provisioning* dell'hardware o del software, o l'installazione e la configurazione sono gestite autonomamente e non sono una responsabilità dello sviluppatore (poiché sono gestite dal servizio automaticamente).

Nel contesto dello stage è stato utilizzato per garantire la persistenza di tutti i dati strutturati dell'applicazione. Apposite funzioni Lambda si sono occupate della creazione, della cancellazione, dell'aggiornamento e del reperimento di questi dati.



Figura 2.7: Logo di Amazon DynamoDB

2.2.4 Amazon S3

Amazon Simple Storage Service (il cui acronimo è Amazon S3) è uno *storage* di oggetti creato per memorizzare e ripristinare qualsiasi volume di dati da qualunque origine: siti web, applicazioni per mobile o dati provenienti da diversi dispositivi. Può essere utilizzato per memorizzare file multimediali ed è ideale per acquisire dati quali foto, video o immagini di risoluzione elevata da dispositivi mobili, backup di dispositivi mobile o computer.

Nel contesto dello stage è stato utilizzato per garantire la persistenza di tutti i dati non strutturati dell'applicazione. Apposite funzioni Lambda si sono occupate della gestione di questi dati.

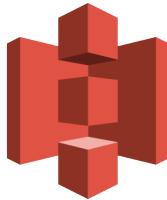


Figura 2.8: Logo di Amazon S3

2.2.5 Amazon Cognito

Amazon Cognito è un servizio per la gestione dell'autenticazione degli utenti. Tramite questo servizio è possibile configurare per una applicazione le opzioni per

la registrazione e anche di autenticare gli utenti direttamente tramite social (come Facebook e Twitter). È anche possibile occuparsi in maniera veloce ed intuitiva dell'aggiunta o della rimozione degli utenti, e anche di associarli a vari gruppi logici ai quali possono essere correlati permessi a diversi livelli.

Amazon Cognito mette a disposizione due elementi principali: lo *User Pool* e le *Federated Identities*.

Lo User Pool non è altro che un insieme (ovvero un gruppo) di utenti associati all'applicazione; è qui che è possibile definire (creare) un utente, la sua appartenenza a uno o più gruppi, gestire questi gruppi e definire degli attributi specifici di alcuni utenti. Le Federated Identities invece sono un servizio che permette l'autenticazione da sistemi totalmente diversi rispetto all'applicazione che si sta sviluppando (tali sistemi potrebbero essere dei *social network* come Facebook, Twitter, ecc...); queste autenticazioni vengono appunto federate in un'unico strato logico riferito a un singolo utente.

Nel contesto dello stage è stato molto utilizzato per implementare tutta la parte del progetto legata alla gestione dell'autenticazione. È stato necessario servirsi solo dello User Pool.



Figura 2.9: Logo di Amazon Cognito

2.3 Tecnologie di supporto

2.3.1 Visual Studio Code

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft e disponibile per i sistemi operativi Windows, macOS e Linux. Gode del supporto a moltissimi linguaggi di vario tipo (dai classici linguaggi di programmazione come C, C++ e Java ai linguaggi di markup come HTML). Possiede un sistema integrato di versionamento del codice (che funziona tramite Git) e un debugger.

È stato scelto come editor di testo principale per la scrittura del codice sorgente del front-end dell'applicazione visto che il suo supporto ai linguaggi TypeScript, JavaScript, HTML, CSS e JSON lo rendeva perfetto per lo sviluppo di applicazioni basate su Angular 4.



Figura 2.10: Logo di Visual Studio Code

2.3.2 Git

Git è un software di controllo di versione distribuito, creato nel 2005 da Linus Torvalds (creatore di Linux). È usabile principalmente da interfaccia linea di comando, ed oggi è in assoluto uno dei software per il controllo di versione più utilizzati. Come tutti i software per il controllo di versione, si basa sul concetto di *repository*, ovvero un ambiente in cui vengono immagazzinati i metadati che possono essere recuperati e aggiornati da chi può averne accesso (è possibile ripristinare versioni precedenti dei dati caricati nel *repository*, poiché essi non vengono sovrascritti con gli aggiornamenti). Un caricamento che effettua un aggiornamento del repository viene chiamato *commit* (e si esegue tramite l'opportuno comando *commit*). Per scaricare in locale una copia del repository invece si utilizza il comando *clone*.

Tutto il codice sorgente creato durante lo stage è stato versionato su un *repository* Git privato, grazie al servizio AWS CodeCommit.



Figura 2.11: Logo di Git

2.3.3 AWS CodeCommit

AWS CodeCommit è un servizio di *hosting* per progetti software ed è una implementazione dello strumento di controllo versione distribuito Git.

L'uso di questo servizio ha permesso di lavorare su un repository Git privato in modo sicuro e performante.



Figura 2.12: Logo di AWS CodeCommit

2.3.4 AWS IAM

AWS Identity and Access Management (conosciuto come AWS IAM), è un servizio per la gestione di identità (IM) che consente di controllare l'accesso a tutti i servizi e tutte le risorse AWS per una certa tipologia di utenti. È infatti possibile creare e gestire utenti o gruppi di utenti utilizzando autorizzazioni a diversi livelli per consentire o negare l'accesso a un insieme di risorse AWS.

L'uso di opportune credenziali AWS IAM si è rivelato fondamentale sia per l'accesso a tutti i servizi AWS del back-end sia per interfacciarsi con il repository Git privato con il servizio AWS CodeCommit (tali credenziali erano richieste ogniqualvolta si voleva effettuare una `clone` del repository oppure aggiornamenti su di esso tramite `commit`).



Figura 2.13: Logo di AWS IAM

2.3.5 Adobe Experience Design

Adobe Experience Design (abbreviato in Adobe XD) è un software di design grafico sviluppato da Adobe per la progettazione di moderne *user experience* su moltissimi tipi di dispositivi. È supportato dai sistemi operativi macOS e Windows 10. Il software è al momento ancora in versione Beta, e pertanto scaricabile gratuitamente da internet.

Questo software è stato utilizzato nell'intera fase di prototipizzazione dell'interfaccia utente (tale fase è descritta dettagliatamente nel paragrafo 4.1) per la creazione dei mockup delle pagine web assegnate di cui era prevista la realizzazione.



Figura 2.14: Logo di Adobe Experience Design

2.3.6 Tema grafico Inspinia

Inspinia è un tema grafico (a pagamento) utile per gestire vari aspetti legati al design. È un tema totalmente *responsive* (infatti con poche righe di codice inserite direttamente nei template HTML permette all'applicazione di adattarsi graficamente in modo automatico al dispositivo con il quale viene visualizzata, senza utilizzare direttamente fogli di stile CSS). Inoltre contiene set completi e preimpostati di icone, forms, dashboards, gallerie interattive, mailbox, calendari e molte altre funzionalità tipiche dei siti e delle applicazioni web.

Questo tema è stato il riferimento principale per la progettazione e la creazione della grafica di tutte le componenti dell'applicazione assegnate.



Figura 2.15: Interfaccia del tema grafico Inspinia

2.3.7 Swagger

Swagger è un insieme di strumenti e di specifiche il cui scopo è semplificare i processi di documentazione di API per servizi web *RESTful*. Il tutto consiste in un file di testo (che può essere per esempio in formato JSON) dove è descritto in modo totalmente dichiarativo come funziona una API con tutti i dettagli di input e output.

Nel contesto dello stage è stato proprio un file Swagger versionato sul repository, che, sulla base di quanto scritto al suo interno, si occupava della configurazione dei dati e dei tipi delle chiamate HTTP di Amazon API Gateway.



Figura 2.16: Logo di Swagger

2.3.8 Slack

Slack è una applicazione di messaggistica istantanea molto utile come strumento per la collaborazione aziendale. Grazie a Slack è possibile organizzare, attraverso dei canali specifici (magari relativi a progetti diversi), la comunicazione di un intero gruppo di lavoro o di parte di esso.

È stato utilizzato prevalentemente come sistema per il coordinamento con le altre persone a lavoro sulle diverse parti dell'applicazione Logistic Performance, ma anche per l'aggiornamento alcuni eventi interni all'azienda.



Figura 2.17: Logo di Slack

Capitolo 3

Analisi dei requisiti

In questo capitolo verranno trattati i casi d'uso e i requisiti delle parti dell'applicazione Logistic Performance assegnate. Nella prima parte vengono illustrati tutti i casi d'uso ad alto livello con relative descrizioni. Nella parte finale viene riportata una lista completa dei requisiti dell'applicazione. Alcuni tra i casi d'uso principali sono associati ad un *Diagramma UML 2.0 dei casi d'uso* riportante lo stesso titolo e codice identificativo. I casi d'uso di più basso livello non sono descritti in dettaglio.

Ogni caso d'uso è definito secondo la seguente struttura:

- * **ID:** Il codice univoco identificativo del caso d'uso;
- * **Nome:** Il titolo del caso d'uso;
- * **Attori:** Indica gli attori principali e secondari del caso d'uso. In tutto il contesto dell'applicazione gli utenti del sistema saranno così classificati:
 - **Utente:** indica l'insieme di tutti gli utenti che possono interagire con l'applicazione (quando nelle descrizioni si usa il termine "Utente" ci si riferisce sempre a qualsiasi tipologia di utente che abbia avuto accesso all'applicazione).
 - **Utente Generico:** indica un generico utente dell'applicazione. Tale utente può solo leggere le informazioni dell'applicazione e non può interagire con essa per modificare dei dati.
 - **Utente Amministratore:** indica un utente con particolari privilegi. Tali privilegi possono riguardare la modifica di dati o la creazione di nuovi utenti nel sistema.
- * **Descrizione:** Riporta una breve descrizione del caso d'uso;
- * **Precondizione:** Specifica le condizioni che sono identificate come vere prima del verificarsi degli eventi del caso d'uso;
- * **Postcondizione:** Specifica le condizioni che sono identificate come vere dopo il verificarsi degli eventi del caso d'uso;
- * **Scenario principale:** Rappresenta il flusso principale degli eventi, ovvero il caso più frequente di utilizzo;

- * **Scenari alternativi:** Descrivono i casi d'uso che non fanno parte del flusso principale degli eventi.

3.1 Pagina di dettaglio di una consegna

La *Pagina di dettaglio di una consegna* è la pagina web il cui scopo principale è proprio quello di visualizzare tutte le proprietà di una particolare consegna. L'utente è arrivato a questa pagina web perché ha selezionato la consegna di proprio interesse all'interno della *Pagina della lista delle consegne*. Una consegna può essere di diverse tipologie: essa può essere stata consegnata con successo, consegnata con ritardo, consegnata con riserva (ovvero contestata) oppure non consegnata. Nei primi tre casi l'utente potrà visualizzare la foto della bolla di consegna scattata dal trasportatore (in caso di consegna contestata l'utente potrà essere in grado di visualizzare altre foto che rappresentano il dettaglio della contestazione), la data e l'ora effettiva dell'avvenuta consegna e anche una geolocalizzazione interattiva indicante la posizione geografica in cui la consegna è avvenuta. Nel caso di consegna non ancora effettuata l'utente non avrà ovviamente accesso a tali informazioni poiché ancora inesistenti.

Un utente amministratore, contrariamente a un utente generico, avrà la possibilità di modificare alcune proprietà della consegna in questione, direttamente dalla pagina web.

3.1.1 UC1: Visualizzazione e modifica della lista di proprietà di una consegna

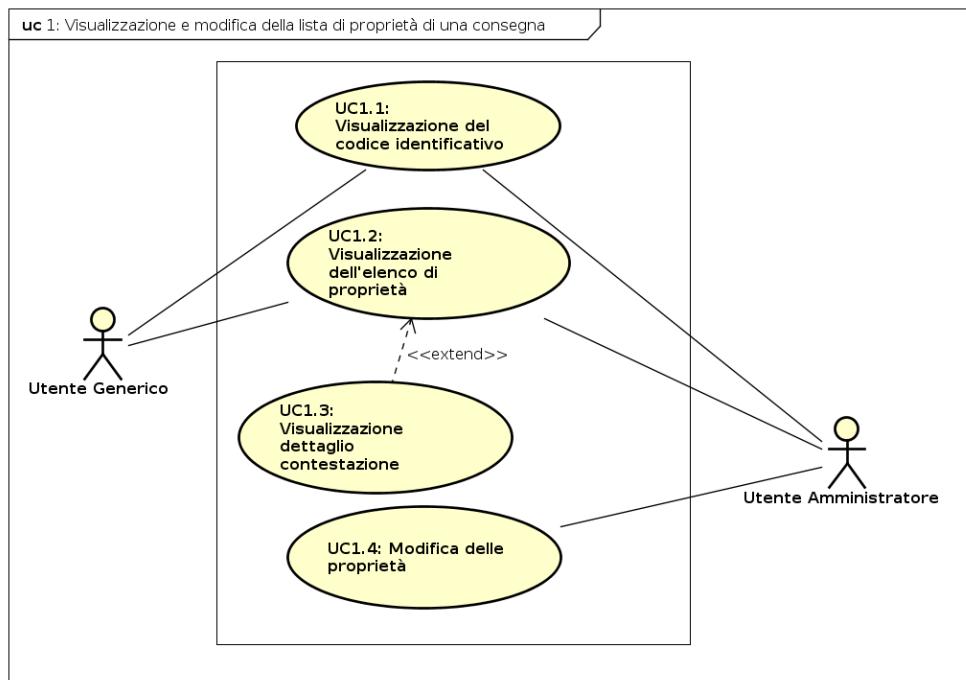


Figura 3.1: UC1: Visualizzazione e modifica della lista di proprietà di una consegna

- * **Attori:** Utente Generico, Utente Amministratore.
- * **Descrizione:** L'Utente ha la possibilità di consultare tutte le proprietà della specifica consegna (incluso l'ID identificativo) a colpo d'occhio. Un Utente Amministratore può modificare alcune di queste proprietà.
- * **Precondizione:** L'Utente ha avuto accesso alla lista di proprietà della specifica consegna perché ha selezionato proprio questa consegna all'interno della *Pagina della lista delle consegne*.
- * **Postcondizione:** L'Utente ha ottenuto informazioni di interesse per la specifica consegna visualizzata. Inoltre, alcune proprietà potrebbero essere state modificate.
- * **Scenario principale:** Qualsiasi Utente può visualizzare la lista di tutte le proprietà della consegna oggetto della pagina.

3.1.2 UC2: Visualizzazione della localizzazione

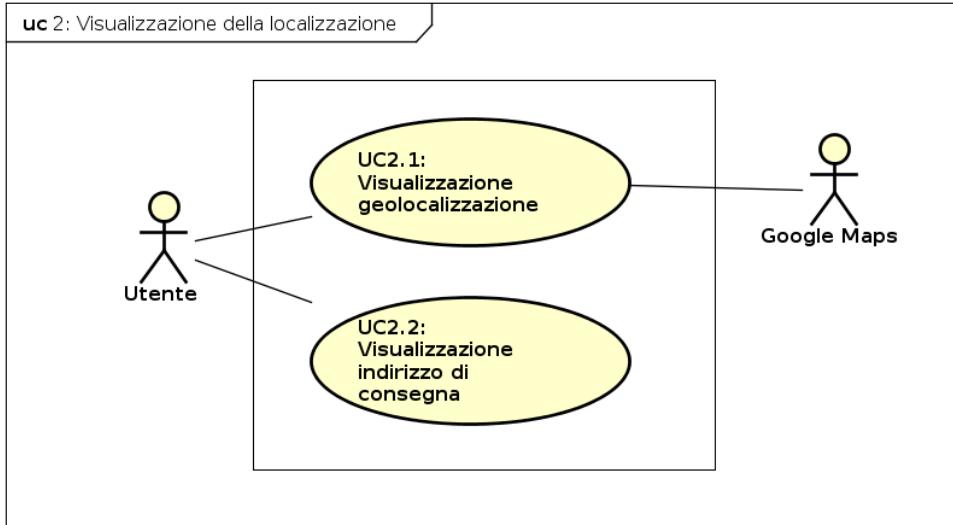


Figura 3.2: UC2: Visualizzazione della localizzazione

- * **Attori:** Utente, Google Maps.
- * **Descrizione:** Qualsiasi Utente dell'applicazione potrà conoscere la localizzazione di una qualsiasi consegna effettuata. Tale localizzazione consiste in una geolocalizzazione interattiva e l'indirizzo preciso in cui la consegna è stata effettuata.
- * **Precondizione:** L'Utente ha avuto accesso alla localizzazione della specifica consegna perché ha selezionato proprio questa consegna all'interno della *Pagina della lista delle consegne*.

- * **Postcondizione:** L'Utente si è reso conto di dove esattamente è stata effettuata la consegna (se è stata effettuata).
- * **Scenario principale:** L'Utente visualizza geolocalizzazione, indirizzo della consegna e una icona esplicativa di geolocalizzazione.
- * **Scenari alternativi:** Geolocalizzazione e indirizzo sono assenti perché la specifica consegna è della tipologia non consegnata; in questo caso ci saranno opportuni placeholder sia al posto della geolocalizzazione che al posto dell'indirizzo che informeranno l'Utente dell'assenza di questi dati. L'icona esplicativa non subirà alcuna modifica.

3.1.3 UC3: Visualizzazione bolla e orario della consegna

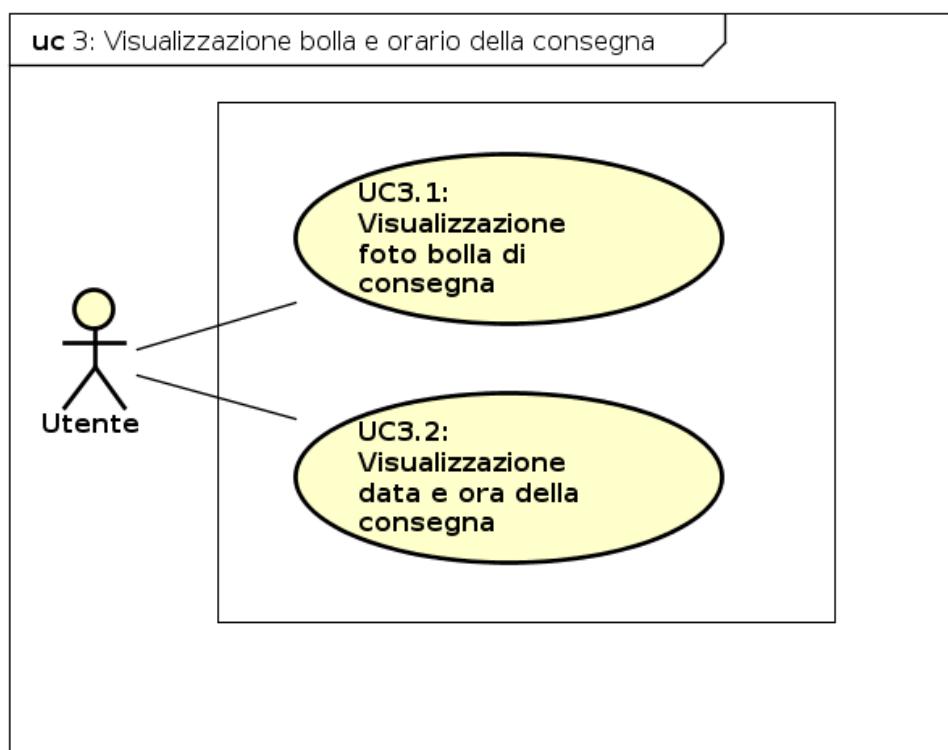


Figura 3.3: UC3: Visualizzazione bolla e orario della consegna

- * **Attori:** Utente.
- * **Descrizione:** Qualsiasi Utente dell'applicazione potrà visualizzare la foto della bolla di consegna (tale fotografia è stata scattata tramite dispositivo mobile dallo specifico trasportatore che si è occupato della consegna). Può anche essere visualizzata la data e l'ora esatta della reale consegna (questo orario corrisponde in realtà esattamente al momento del caricamento della foto dal dispositivo del trasportatore al back-end dell'applicazione).

- * **Precondizione:** L'Utente ha avuto accesso alla bolla di consegna e all'orario della specifica consegna perché ha selezionato proprio questa consegna all'interno della *Pagina della lista delle consegne*.
- * **Postcondizione:** L'Utente ha visualizzato orario della consegna e foto della bolla (se la consegna è stata effettuata).
- * **Scenario principale:** L'Utente può visualizzare la foto della bolla della consegna e leggere l'orario in cui tale consegna è stata effettuata.
- * **Scenari alternativi:** Foto della bolla e orario di consegna sono assenti perché la specifica consegna è della tipologia non consegnata; in questo caso ci saranno opportuni placeholder sia al posto della foto che al posto dell'orario che informeranno l'Utente dell'assenza di questi dati. L'icona esplicativa non subirà alcuna modifica.

3.1.4 UC4: Navigazione

- * **Attori:** Utente.
- * **Descrizione:** L'Utente può spostarsi da questa pagina alle altre pagine dell'applicazione utilizzando la barra di navigazione. Tale barra aiuta l'Utente a evitare il disorientamento e a fargli sempre ricordare in quale punto logico dell'applicazione si trova.¹.
- * **Precondizione:** L'Utente si trova nella *Pagina di dettaglio di una consegna*.
- * **Postcondizione:** L'Utente è tornato indietro alla *Pagina della lista delle consegne*, che è la pagina in cui era stato precedentemente, oppure in un'altra pagina dell'applicazione la cui collocazione era logicamente prima della *Pagina di dettaglio di una consegna*.
- * **Scenario principale:** L'Utente può navigare tra le pagine dell'applicazione visitate in precedenza attraverso la barra di navigazione. Può anche semplicemente visualizzarla senza interagire con essa per rendersi conto di dove è collocata attualmente la pagina.

3.2 Pagina della lista degli utenti

La *Pagina della lista degli utenti* è una pagina web dell'applicazione che è accessibile solo ed esclusivamente da un utente amministratore. Essa mostra una tabella dinamica contenente l'elenco effettivo di tutti gli utenti registrati (ogni riga della tabella corrisponde ad un utente, e nelle colonne si trovano le principali proprietà di questi utenti) e memorizzati nel back-end dell'applicazione grazie al servizio Amazon Cognito. Questa pagina non serve soltanto per dare una panoramica del numero di utenti, ma anche per permettere all'amministratore di osservare il dettaglio di un singolo utente (ovvero tutto l'insieme delle sue proprietà) selezionandolo direttamente da questa tabella (questa selezione farà aprire la *Pagina di dettaglio di un singolo utente*). Questa pagina conterrà anche un bottone che se cliccato dall'amministratore

¹UC4 è stato incluso in questo paragrafo relativo alla *Pagina di dettaglio di una consegna* perché originariamente era stato assegnato come caso d'uso per questa pagina. In realtà tale funzionalità ha riguardato poi tutte le componenti assegnate.

lo farà accedere alla pagina di creazione di un nuovo utente dell'applicazione, ovvero la *Pagina di creazione di un nuovo utente*.

Il template di questa pagina sarà identico a quello della *Pagina della lista delle consegne*, che è una pagina web i cui requisiti non sono stati assegnati direttamente in stage ma su cui è stato comunque effettuato del lavoro in quanto collegata logicamente alla *Pagina di dettaglio di una consegna*.

3.2.1 UC5: Visualizzazione della lista degli utenti

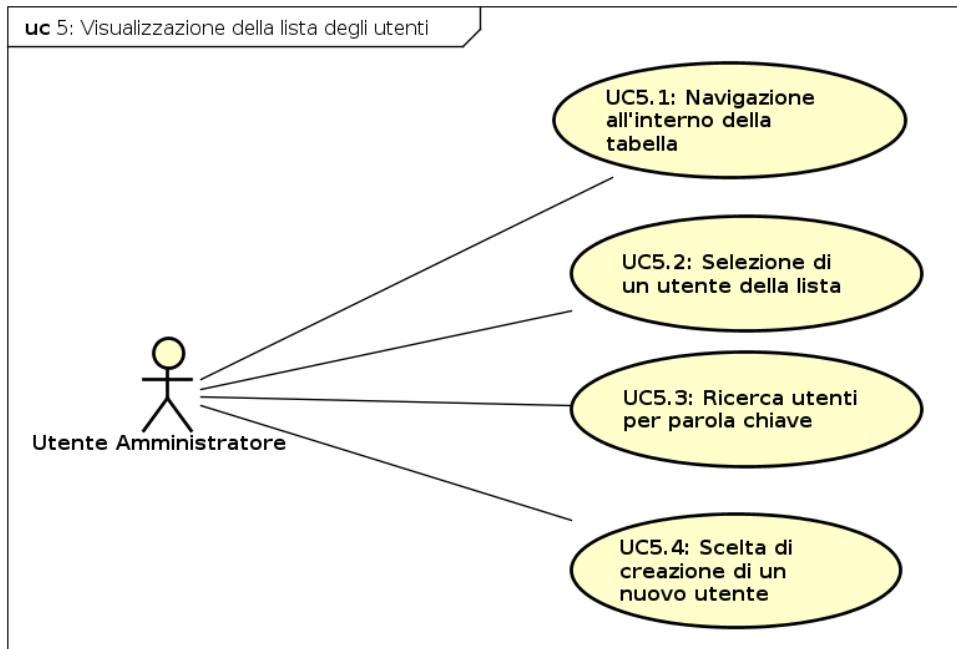


Figura 3.4: UC5: Visualizzazione della lista degli utenti

- * **Attori:** Utente Amministratore.
- * **Descrizione:** L'Utente Amministratore è in grado di visualizzare e interagire con una tabella in cui sono elencati tutti gli utenti registrati dell'applicazione.
- * **Precondizione:** L'Utente Amministratore è l'unico utente che può visualizzare la sezione dell'applicazione *Users Management* direttamente nel menu di navigazione: deve aver fatto click su questa sezione per visualizzare questa pagina.
- * **Postcondizione:** L'Utente Amministratore ha ottenuto informazioni di interesse riguardanti gli utenti dell'applicazione.
- * **Scenari principali:** L'Utente Amministratore ha la possibilità di navigare all'interno di questa lista degli utenti (scorrendo insieme di righe con appositi pulsanti qualora esse dovessero superare un certo limite fissato), selezionare uno qualsiasi di questi utenti, effettuare una ricerca per parola chiave tramite appositi campi di input testuale collocati in cima a ogni colonna della tabella oppure scegliere di creare un nuovo utente dell'applicazione.

- * **Scenari alternativi:** L'Utente Amministratore ha deciso di uscire dalla pagina ed è tornato alla Home dell'applicazione dopo aver fatto click sull'apposito link di ritorno.

3.3 Pagina di dettaglio di un singolo utente

La *Pagina di dettaglio di un singolo utente* è una pagina web dell'applicazione che è accessibile solo ed esclusivamente da un utente amministratore. L'amministratore è arrivato a questa pagina web perché ha selezionato l'utente di proprio interesse all'interno della *Pagina della lista degli utenti*. Questa pagina è associata ad un particolare utente dell'applicazione e mostra semplicemente un elenco di proprietà che lo riguardano (come per esempio lo stato, la data di creazione, la mail di registrazione ecc...). Viene anche data la possibilità di modificare i gruppi a cui l'utente appartiene (questi sono gruppi logici definiti a livello applicativo), eliminandoli o inserendoli direttamente da una funzionalità esposta dal front-end.

3.3.1 UC6: Visualizzazione e modifica della lista di proprietà di un utente

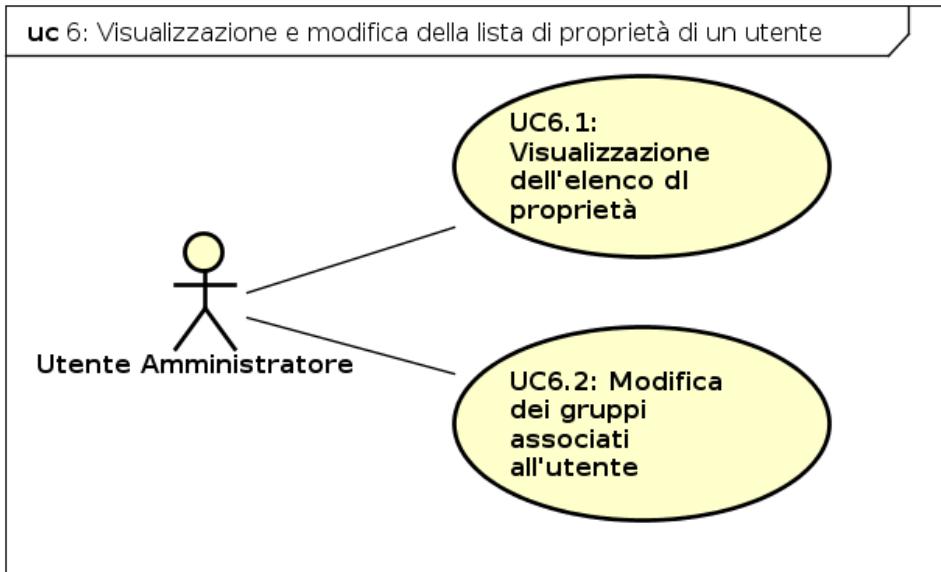


Figura 3.5: UC6: Visualizzazione e modifica della lista di proprietà di un utente

- * **Attori:** Utente Amministratore.
- * **Descrizione:** L'Utente Amministratore può consultare tutte le proprietà di uno specifico utente a colpo d'occhio. Può anche decidere a quali gruppi dell'applicazione l'utente appartiene (l'utente può appartenere a zero o più gruppi).

- * **Precondizione:** L'Utente Amministratore ha avuto accesso alla lista di proprietà dello specifico utente perchè ha selezionato proprio questo utente all'interno della *Pagina della lista degli utenti*.
- * **Postcondizione:** L'Utente Amministratore ha ottenuto informazioni di interesse per lo specifico utente visualizzato. Inoltre, l'insieme dei gruppi a cui lo specifico utente apparteneva potrebbe essere stato modificato.
- * **Scenario principale:** L'Utente Amministratore ha la possibilità di visualizzare l'intera lista di proprietà di un utente. L'Utente Amministratore può modificare in modo dinamico l'insieme dei gruppi a cui appartiene il determinato utente oggetto della pagina.
- * **Scenari alternativi:** L'Utente Amministratore ha deciso di uscire dalla pagina ed è tornato alla *Pagina della lista degli utenti* dell'applicazione dopo aver fatto click sull'apposito link di ritorno.

3.4 Pagina di creazione di un nuovo utente

La *Pagina di creazione di un nuovo utente* è una pagina web dell'applicazione che è accessibile solo ed esclusivamente da un utente amministratore. L'amministratore è arrivato a questa pagina web perchè ha cliccato l'apposito pulsante per effettuare la creazione di un nuovo utente nella *Pagina della lista degli utenti*. Questa pagina ha all'interno una grande form che contiene una serie di campi di input, precisamente per inserire username (unico campo obbligatorio per la creazione di un nuovo utente), password, email, numero di telefono e per selezionare se la notifica della creazione debba essere spedita per email o per SMS (quest'ultima funzionalità è resa possibile grazie ad Amazon Cognito).

3.4.1 UC7: Creazione di un nuovo utente

- * **Attori:** Utente Amministratore.
- * **Descrizione:** L'Utente Amministratore può inserire i dati negli appositi campi di input della form per creare un nuovo utente dell'applicazione.
- * **Precondizione:** L'Utente Amministratore ha avuto accesso alla form per la creazione di un nuovo utente perchè ha cliccato l'apposito pulsante per la creazione di un nuovo utente nella *Pagina della lista degli utenti*.
- * **Postcondizione:** L'Utente Amministratore ha creato un nuovo utente e le proprietà di questo utente sono le stesse inserite nei campi di input della form.
- * **Scenario principale:** Un nuovo utente può essere creato a partire da inserimento di dati in campi di input obbligatori. Se i dati obbligatori non sono stati inseriti sarà impossibile per l'Utente Amministratore creare un nuovo utente. L'Utente Amministratore può anche inserire dati opzionali (cioè che non influiscono sulla possibilità di creazione di un utente) nei campi di input opzionali.
- * **Scenari alternativi:** L'Utente Amministratore ha inserito dei dati opzionali non consentiti che hanno generato un errore che ha impedito la creazione di un nuovo utente. In questo caso l'Utente Amministratore viene informato del fatto che c'è stato un errore e anche della tipologia di errore che si è verificata.

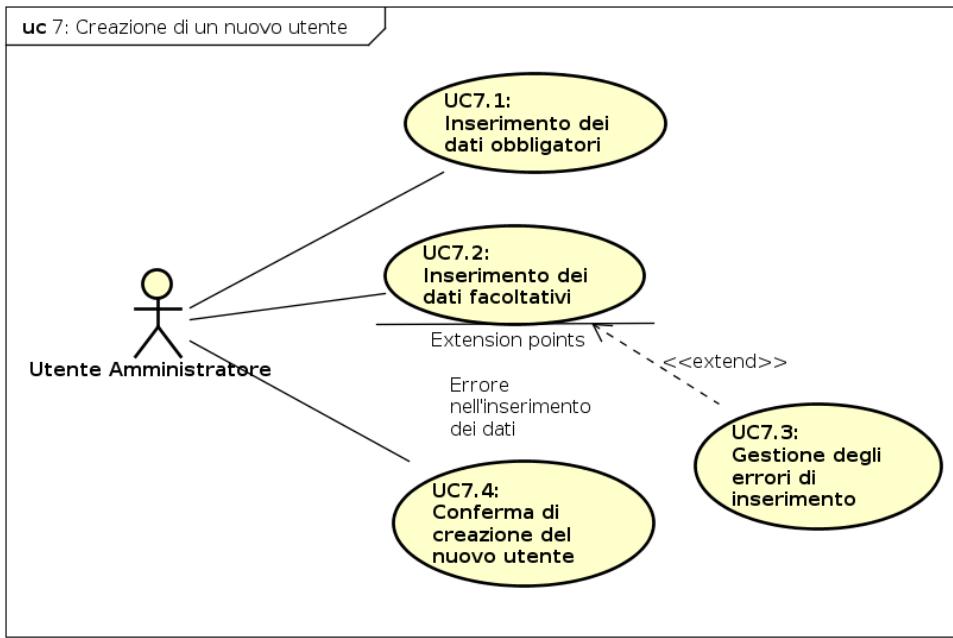


Figura 3.6: UC7: Creazione di un nuovo utente

3.5 Requisiti

In questa sezione viene riportato un elenco dei principali requisiti individuati a partire dai casi d'uso raccolti nelle sezioni precedenti riguardanti le quattro pagine web dell'applicazione *Logistic Performance* assegnate. Ad ogni requisito viene assegnato un **codice identificativo** per semplificarne il tracciamento.

Tale codice è una sigla che rispetta la seguente notazione:

R[Tipo][Rilevanza strategica]-[Numero univoco]

Tipo:

- * **F:** Requisiti funzionale (descrivono le funzionalità e i servizi nel prodotto).
- * **N:** Requisiti non funzionali (definiscono vincoli sullo sviluppo del prodotto ma non sulle funzionalità).

Rilevanza strategica:

- * **O:** Requisito obbligatorio (sono esplicitamente richiesti dal cliente).
- * **D:** Requisiti desiderabili (non strettamente necessari ma con un buon valore aggiunto riconoscibile).
- * **F:** Requisiti opzionali (relativamente utili o contrattabili in seguito con il cliente).

Seguiranno due tabelle. La prima elencherà i principali requisiti funzionali (ognuno di essi è scaturito unicamente da un caso d'uso), la seconda elencherà i requisiti non funzionali citandone le fonti (tipicamente questi sono stati suggeriti dal Tutor Aziendale in fase di riunione).

Tabella 3.1: Tabella dei Requisiti Funzionali

Requisito	Descrizione	Use Case
RFO-1	La consegna ha per forza una delle 4 tipologie: Consegnata con successo, Consegnerà con riserva, Consegnerà con ritardo o Non consegnata	UC1
RFO-2	Un utente è arrivato alla <i>Pagina di dettaglio di una consegna</i> perchè ha cliccato la relativa consegna nella tabella della <i>Pagina della lista delle consegne</i>	UC1
RFO-3	Il codice identificativo di una consegna è univoco	UC1
RFO-4	Nel caso di Consegna con riserva, deve essere visualizzato il dettaglio della contestazione	UC1
RFD-5	La casella di input che consente la modifica ha come valore di default una stringa placeholder contenente il valore del campo dati di cui si sta effettuando la modifica	UC1
RFD-6	Solo un utente amministratore può modificare alcuni dati di una consegna	UC1
RFD-7	Una proprietà deve essere espressa nella forma <i>icona-valore-chiave</i>	UC1
RFD-8	Al click sulla foto del dettaglio di una riserva si apre una lightbox che mostra la foto nelle sue dimensioni originali, che l'utente può richiudere quando vuole	UC1
RFF-9	La modifica di un utente è implementata grazie al click di una icona visibile tramite hover	UC1
RFF-10	Le fotografie del dettaglio della riserva possono essere più di una	UC1
RFF-11	La foto del dettaglio di una riserva deve apparire sotto la lista di proprietà di una consegna	UC1
RFO-12	L'applicazione di Google Maps integrata funziona ed è centrata nelle giuste coordinate GPS in cui è stata scattata la foto della bolla	UC2
RFO-13	Viene visualizzata una immagine placeholder al posto dell'applicazione di Google Maps nel caso in cui non siano disponibili le coordinate GPS	UC2
RFO-14	Viene visualizzato l'indirizzo di destinazione se la consegna è stata effettuata	UC2
RFO-15	Viene visualizzata una stringa placeholder al posto dell'indirizzo se non sono disponibili le coordinate GPS	UC2
RFD-16	Una icona rappresentativa fa intuire all'utente che la componente che sta visualizzando si riferisce a una geolocalizzazione	UC2
RFO-17	Viene visualizzata la foto della bolla di consegna se presente	UC3
RFO-18	Viene visualizzata una immagine placeholder al posto della foto della bolla di consegna nel caso in cui essa non sia disponibile	UC3
RFO-19	Viene visualizzato l'orario dell'avvenuta consegna se la bolla di consegna è stata fotografata	UC3
RFO-20	Viene visualizzata una stringa placeholder al posto dell'orario se non è disponibile la foto della bolla della consegna	UC3
RFD-22	Al click sulla foto della bolla di consegna si apre una lightbox che mostra la foto nelle sue dimensioni originali, che l'utente può richiudere quando vuole	UC3
RFD-23	Una icona rappresentativa fa intuire all'utente che la componente che sta visualizzando si riferisce a una fotografia	UC3

Requisito	Descrizione	Use Case
RFO-24	La navigazione deve essere presente in tutte le pagine assegnate (Pagina di dettaglio di una consegna, Pagina della lista degli utenti, Pagina di creazione di un nuovo utente, Pagina di dettaglio di un singolo utente)	UC4
RFD-25	La navigazione deve essere implementata tramite una <i>breadcrumb</i> dinamica e non statica	UC4
RFO-26	Al click di una riga nella tabella degli utenti, si apre la <i>Pagina di dettaglio di un singolo utente</i> relativa all'utente individuato dalla riga selezionata	UC5
RFO-27	È possibile cercare gli utenti per parole chiave	UC5
RFO-28	Al click sul bottone per la creazione di un nuovo utente si viene renderizzati alla <i>Pagina di creazione di un nuovo utente</i>	UC5
RFD-29	Se troppo lunga, la tabella può essere divisa in facciate navigabili attraverso un pannello di navigazione	UC5
RFD-30	Se il cursore passa sopra una riga della tabella, essa viene messa in evidenza rispetto alle altre	UC5
RFO-31	Deve essere possibile l'associazione e la dissociazione di qualsiasi gruppo già esistente rispetto a uno specifico utente	UC6
RFO-32	Al click sul link di ritorno l'utente viene renderizzato alla <i>Pagina della lista degli utenti</i>	UC6
RFD-33	Il titolo della <i>Pagina di dettaglio di un singolo utente</i> è il nome identificativo dell'utente a cui fa riferimento la pagina	UC6
RFD-34	Una riga che rappresenta una proprietà statica di un utente è nel formato <i>icona-valore-chiave</i>	UC6
RFD-35	Le icone scelte per rappresentare un dato obbligatorio di un utente sono rappresentative del dato in questione	UC6
RFD-36	L'icona scelta per rappresentare un dato non obbligatorio è sempre la stessa	UC6
RFF-37	La modifica dei gruppi associati allo specifico utente viene implementata con una casella di input a selezione multipla	UC6
RFO-38	L'inserimento dell'username è obbligatorio per la creazione di un nuovo utente	UC7
RFO-39	Un nuovo utente non viene creato se è stata violata una condizione nell'inserimento dei dati (numero di telefono senza prefisso, lunghezza della password errata, username già esistente, formato per la mail non valido)	UC7
RFO-40	Al click sul link di ritorno l'utente viene renderizzato alla <i>Pagina della lista degli utenti</i>	UC7
RFD-41	Non sarà possibile premere il pulsante per creare un nuovo utente se non è stato inserito un username	UC7
RFD-42	Se il nuovo utente non viene creato, l'utente viene informato dell'errore avvenuto	UC7
RFD-43	Se il nuovo utente viene creato, l'utente viene informato dell'avvenuta creazione del nuovo utente	UC7

Tabella 3.2: Tabella dei Requisiti Non Funzionali

Requisito	Descrizione	Fonti
RNO-1	Lo stile grafico delle pagine deve essere <i>responsive</i>	Verbale 27-07-2017
RNO-2	L'interazione con il back-end per quanto riguarda l'implementazione della <i>Pagina di dettaglio di una consegna</i> deve essere fatta utilizzando il servizio Amazon API Gateway	Verbale 27-07-2017
RND-3	Lo stile grafico delle pagine deve essere fatto seguendo il più possibile il tema grafico Inspinia	Verbale 27-07-2017
RND-4	È opportuno ridurre al minimo il codice sorgente dei template HTML, cercando di automatizzare il più possibile la struttura delle pagine, servendosi magari di specifiche classi logiche create ad hoc	Verbale 04-09-2017
RND-5	La lavorazione sulle pagine web deve seguire piccoli cicli di sviluppo piuttosto che un unico grande ciclo	Verbale 27-07-2017
RND-6	Tutti i test svolti devono essere forniti in modo comprensibile e devono essere completi.	Verbale 22-08-2017
RND-7	L'intero progetto deve essere accompagnato da documentazione completa	Piano di lavoro
RNF-8	L'interazione con il back-end per quanto riguarda i dati di Amazon Cognito deve essere fatta usando l'SDK JavaScript	Verbale 18-09-2017

Capitolo 4

Progettazione e sviluppo

Questo capitolo si concentra sull'intera fase di progettazione delle pagine web di Logistic Performance. Verrà descritto nel dettaglio innanzitutto il processo di prototipizzazione dell'interfaccia utente per tutte le pagine web assegnate, poi verranno presentate la progettazione architetturale e la progettazione di dettaglio.

4.1 Prototipizzazione dell'interfaccia utente

In questa sezione verrà illustrata la graduale progettazione della grafica e del funzionamento generale delle pagine web dell'applicazione Logistic Performance assegnate delle quali sono stati analizzati i requisiti. Ogni scelta progettuale in questa fase di prototipizzazione non è casuale ma deriva da uno o più requisiti scaturiti dalla fase di analisi (descritta dettagliatamente nel Capitolo 3). I mockup sono divisi in quattro livelli di astrazione progressivi, che contengono via via informazioni sempre più dettagliate:

- * **Wireframe:** A questo livello progettuale interessa soltanto stabilire dove saranno collocate e quanto saranno approssimativamente grandi le aree che si è deciso di occupare. Ciascuna di queste aree avrà un compito ben preciso, che tipicamente corrisponderà a una macrofunzionalità dell'applicazione (questo tipo di progettazione riguarda esclusivamente lo spazio all'interno della pagina web).
- * **Wireframe con dettaglio delle aree:** A questo livello si progetta il dettaglio delle aree individuate nella fase precedente: per ciascuna area si definirà la tipologia dei dati contenuti in essa, oltre che la loro specifica posizione all'interno della stessa.
- * **Mockup:** In questa fase progettuale si finisce definitivamente di definire la tipologia (logica) dei dati visualizzati descrivendone le funzionalità e si comincia ad abbozzare il layout grafico della pagina web.
- * **Mockup della User Interface:** In questa fase finale si cerca di rendere il mockup quasi una copia di ciò che sarà il risultato grafico finale vero e proprio. A questo livello progettuale ci si concentra solamente sulla *user experience* e quindi su concetti di usabilità e gestione dell'informazione, e non più su aspetti legati alla logica applicativa.

4.1.1 Pagina di dettaglio di una consegna

Wireframe

A partire dai requisiti raccolti per la *Pagina di dettaglio di una consegna*, è stato scelto di creare 4 diverse aree, una per ogni funzionalità individuata, più l'area per il menu di navigazione (l'implementazione di quest'ultima area non ha mai fatto parte dei requisiti in quanto funzionalità che riguardava l'intera applicazione).

È stato scelto uno schema a **Layout a F**, che tipicamente favorisce una *user experience* efficace. I contenuti vengono disposti “a forma di F”; poichè gli utenti solitamente prima leggono in senso orizzontale da sinistra verso destra, attraversando la parte superiore della pagina. Per quanto riguarda la “Barra di navigazione”, essa è stata posta nella parte orizzontale della pagina web, che è proprio il luogo in cui per convenzione viene solitamente implementata la navigazione. Le aree “Lista di proprietà” e “Localizzazione” sono state ritenute essere più importanti per un utente di questa applicazione piuttosto che l'area “Bolla e orario della consegna” in quanto più informative (è per questo motivo che “Lista di proprietà” occupa praticamente la metà sinistra del layout mentre “Localizzazione” è stata collocata nella parte destra ma più in alto rispetto a “Bolla e orario della consegna”).



Figura 4.1: Wireframe di *Pagina di dettaglio di una consegna*

Wireframe con dettaglio delle aree

La lista di proprietà inizierà con il titolo identificativo (che sarà l'ID della specifica consegna) e accanto una icona eplicativa della tipologia della consegna. Il resto della lista di proprietà sarà un elenco di elementi nel formato *icona-valore-chiave* e saranno allineate in forma tabellare con lunghezza predefinita.

Nonostante in caso di consegna con riserva la foto con il dettaglio della contestazione è prevista essere una sola, si è pensato di implementare questa funzionalità come una galleria di immagini (che avrà ovviamente inizialmente un solo valore) collocata subito sotto l'elenco delle proprietà. In futuro potrebbe esistere il caso in cui saranno disponibili più foto di contestazione per una medesima consegna con riserva.

Le due aree dedicate a “Localizzazione” e a “Bolla e orario della consegna” sono state definite nel medesimo modo: uno spazio grande della stessa misura per le funzionalità di geolocalizzazione e visualizzazione della foto della bolla (nel caso in cui manchino queste informazioni gli spazi saranno occupati da immagini placeholder della stessa misura) e uno spazio sottostante per le icone rappresentative delle funzionalità e le

stringhe di testo (anch'essi della stessa misura).

È stato scelto infine di progettare la barra di navigazione con una tipica breadcrumb.

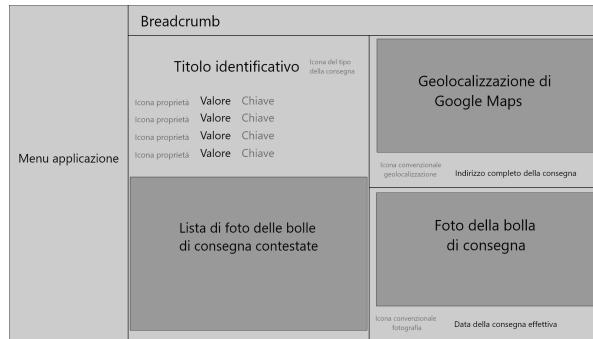


Figura 4.2: Wireframe con dettaglio delle aree di *Pagina di dettaglio di una consegna*

Mockup

È stato scelto di dare una buona rilevanza alla diversificazione tra tipologie distinte della consegna; per fare ciò si è pensato di cambiare la colorazione del titolo della consegna (corrispondente al suo numero identificativo) a seconda della sua tipologia: questo garantisce un riconoscimento della tipologia della consegna a colpo d'occhio da parte di un utente. Se tale utente dovesse utilizzare l'applicazione per più di una volta (e questo è proprio il caso dell'utenza di Logistic Performance) imparerebbe in poco tempo a distinguere le quattro colorazioni associate alle quattro diverse tipologie di consegna. Anche l'icona accanto al titolo cambia (e ciò garantisce un riconoscimento della diversa tipologia di consegna anche a persone affette da disturbi come il daltonismo, migliorando l'accessibilità della pagina).

Le icone scelte (così come le immagini placeholder) sono state pensate per essere più intuitive possibili (per esempio il guidatore sarà rappresentato con il classico omino, la data prevista con il calendario e la foto della bolla con una macchina fotografica) e sono coerenti con le icone scelte per la versione mobile dell'applicazione (questo migliora l'usabilità dell'applicazione).

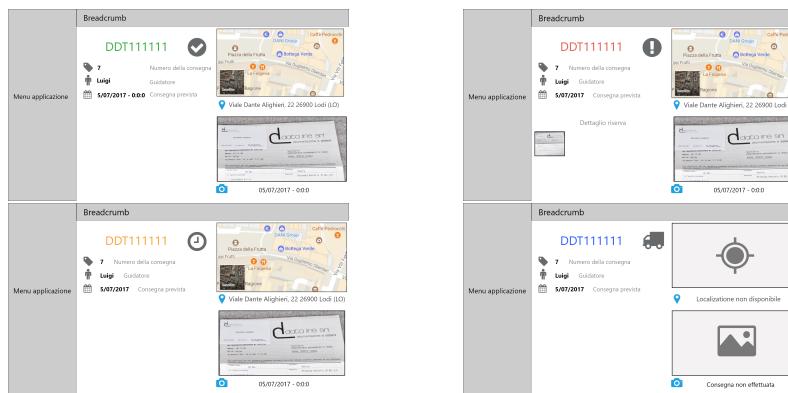


Figura 4.3: Mockup di *Pagina di dettaglio di una consegna*

Mockup della User Interface

I colori, i font, e gli stili del menu e della breadcrumb sono stati scelti coerentemente con quelli messi a disposizione dal tema grafico Inspinia. Essendo l’inglese la lingua dell’applicazione Logistic Performance, tutte le scritte sono state tradotte dall’italiano all’inglese.

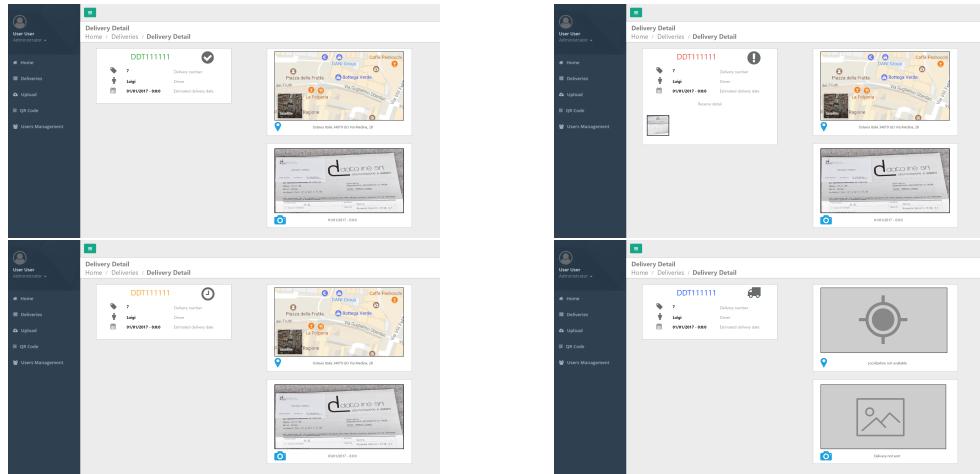


Figura 4.4: Mockup della User Interface di *Pagina di dettaglio di una consegna*

4.1.2 Pagina della lista degli utenti

Wireframe

Prendendo ispirazione dalla *Pagina della lista delle consegne*, è stato ritenuto opportuno progettare la tabella degli utenti dell’applicazione in modo che occupasse la pagina web per intero. Per questo motivo, l’area “Tabella degli utenti” occupa interamente lo spazio della pagina web (tralasciando la solita “Barra di navigazione” e “Menu applicazione” che hanno una collocazione fissa in tutta l’applicazione).



Figura 4.5: Wireframe di *Pagina della lista degli utenti*

Wireframe con dettaglio delle aree

L'intera "Tabella degli utenti" sarà collocata al centro della pagina e avrà esattamente cinque colonne (ciascuna per ogni proprietà obbligatoria di un utente, identificata da una specifica "Chiave") e un numero variabile di righe (ce ne sarà una per ogni utente memorizzato). La ricerca degli utenti per parole chiave sarà implementata grazie a cinque caselle di input (ognuna posta in cima a ciascuna colonna): in questo modo sarà possibile ricercare un utente in base ad un valore effettivo di una sua specifica proprietà. In alto a sinistra è stato scelto di inserire il bottone per l'inserimento di un nuovo utente: questa scelta è stata fatta per evitare che con il crescere e il diminuire delle righe della tabella il pulsante si spostasse, disorientando così l'utente (la funzionalità di questo pulsante è infatti del tutto slegata da quella della tabella).

Infine, le informazioni per la navigazione all'interno della tabella stessa sono state collocate in fondo a sinistra subito sotto l'ultima riga e al contrario del bottone per la creazione di un nuovo utente la loro posizione cambierà con il crescere e il diminuire delle righe della tabella (questo dovrebbe garantire che l'utente riesca a capire a colpo d'occhio se le righe della tabella sono finite o se ne esistono altre che possono essere visualizzate usufruendo di questa navigazione).

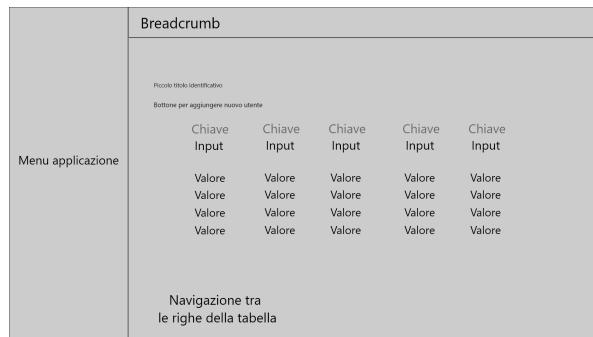


Figura 4.6: Wireframe con dettaglio delle aree di *Pagina della lista degli utenti*

Mockup

Le cinque caselle di input per la ricerca per parola chiave conterranno una stringa placeholder (ciascuna riferita all'apposita proprietà) che informerà l'utente sulla tipologia di ricerca per parola chiave.

È stato scelto che il numero di righe della tabella cambierà dinamicamente durante l'inserimento di testo all'interno di una qualsiasi casella di input filtrando fin da subito gli utenti le cui proprietà sono coerenti con la ricerca effettuata ¹. I valori di tipo data saranno visualizzabili nel formato "GG:MM:YYYY - H:M:S" e la ricerca nelle apposite caselle di input richiederà di inserire un dato nel formato "YYYY-MM-GG".

La barra di navigazione è stata progettata per variare dinamicamente in base al numero di facciate di una tabella. Una facciata è stata definita come una tabella di otto righe. Se gli utenti dell'applicazione dovessero essere più di otto, le successive righe della tabella appariranno in un'altra facciata, per cui ci sarà un numero di facciate pari al

¹Per esempio se si vorrà cercare l'utente "Mario" e in tabella sono presenti solo gli utenti "Martino", "Michelangelo" e "Antonio", con l'inserimento della lettera "M" nella colonna Username, in tabella appariranno soltanto "Martino" e "Michelangelo", poiché entrambi iniziano con "M". Se la prossima lettera in input sarà una "a" nella tabella rimarrà visibile soltanto la riga riferita a "Martino")

numero di utenti diviso otto più eventualmente un'altra facciata per gli utenti rimanenti. Questo numero di facciate è indicato in ogni momento nella barra di navigazione in fondo alla pagina, in modo che l'utente possa sempre rendersi conto del numero di facciate della tabella (Una stringa sotto la barra informerà l'utente circa la posizione della facciata visualizzata rispetto al numero totale delle facciate).

Utenti di Logistic Performance					
Nuevo utente					
Username	Enabled	Status	Updated	Created	
user1	true	CONFIRMED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user2	true	CONFIRMED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user3	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user4	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user5	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user6	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user7	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user8	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	

Figura 4.7: Mockup di Pagina della lista degli utenti

Mockup della User Interface

Lo stile del bottone per la creazione di un nuovo utente e lo stile della barra per la navigazione tra le righe della tabella sono coerenti con quelli messi a disposizione dal tema grafico Inspinia. Essendo l'inglese la lingua dell'applicazione Logistic Performance, tutte le scritte sono state tradotte dall'italiano all'inglese. Inoltre, essendo stata progettata la ricerca dinamica delle date nelle caselle di input in modo che esse ricevessero un inserimento testuale del tipo "YYYY-MM-GG", si è pensato di adattare la stringa placeholder di questi campi in modo da informare l'utente sulle modalità di inserimento di una data valida. Infine si è scelto di implementare un meccanismo di *hover* per la selezione di un singolo utente. Essendo infatti le righe della tabella cliccabili, un passaggio del mouse su una qualsiasi riga della tabella la metterà in evidenza rispetto alle altre righe (colorandola di grigio): in questo modo l'utente capirà immediatamente che potrà interagire con quella determinata riga. Questo migliora la *user experience*.

Users Management					
Home / Users Management					
Users List					
Create New User					
Username	Enabled	Status	Updated	Created	
user1	true	CONFIRMED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user2	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user3	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user4	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user5	true	RESET_REQUIRED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user6	true	CONFIRMED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user7	true	CONFIRMED	01/01/2017 - 0:00	01/01/2017 - 0:00	
user8	true	CONFIRMED	01/01/2017 - 0:00	01/01/2017 - 0:00	

Figura 4.8: Mockup della User Interface di Pagina della lista degli utenti

4.1.3 Pagina di dettaglio di un singolo utente

Wireframe

L'area "Lista di proprietà" occuperà per intero la pagina web in quanto la *Pagina di dettaglio di un singolo utente* contiene per definizione soltanto un elenco di proprietà di un utente dell'applicazione.

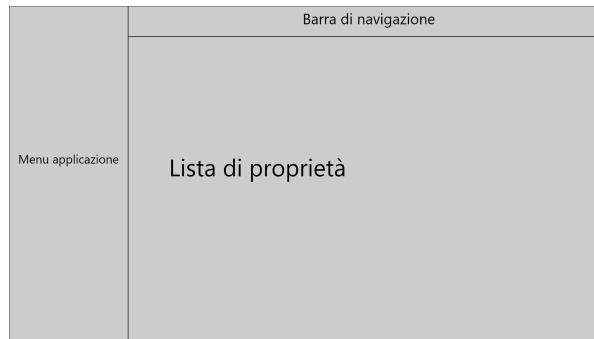


Figura 4.9: Wireframe di *Pagina di dettaglio di un singolo utente*

Wireframe con dettaglio delle aree

La progettazione di quest'area è molto simile a quella della lista delle proprietà della *Pagina di dettaglio di una consegna*.

L'elenco di proprietà conterrà un numero variabile di campi dati (alcuni campi dati sono obbligatori e quindi saranno sempre in numero fisso, il resto saranno facoltativi e il loro numero non sarà costante) e per questo motivo la sua lunghezza non sarà sempre uguale. Ciononostante non è stata implementata una navigazione in base al numero di facciate (come fatto nella *Pagina della lista degli utenti*) poichè i dati facoltativi sono un numero ridotto e limitato.

Il nome dell'utente apparirà come il titolo della pagina, mentre la lista sarà un elenco di elementi nel formato *icona-valore-chiave* che saranno allineati in forma tabellare con lunghezza predefinita. È stato inoltre previsto un link di ritorno alla *Pagina della lista degli utenti* in posizione fissa collocato in basso a sinistra.



Figura 4.10: Wireframe con dettaglio delle aree di *Pagina di dettaglio di un singolo utente*

Mockup

La distanza fissa tra il valore e la chiave è stata progettata in modo tale che la stringa contenente il valore con la massima lunghezza (questo è un valore fissato e non variabile) non si sovrapponga mai con la stringa della chiave.

Le icone che identificano le proprietà obbligatorie sono state pensate per essere più intuitive possibili (per esempio un gruppo sarà rappresentato con una icona di un gruppo di omini). Per quanto riguarda le proprietà opzionali (in numero variabile ma limitato) è stato scelto di usare per tutte le stessa icona (quella di una matita in una casella) in modo tale che l'utente a colpo d'occhio capisca quali sono le proprietà obbligatorie e quali invece quelle opzionali.

I valori di tipo data saranno visualizzabili nel formato “GG:MM:YYYY - H:M:S”.

Per quanto riguarda la modifica dei gruppi di appartenenza dell'utente (ovvero l'unica funzionalità di modifica permessa per questa pagina) è stato pensato di implementare la funzionalità tramite una casella di input a selezione multipla preimpostata: in questa casella compariranno in ogni momento tutti i gruppi dell'applicazione a cui è associato l'utente; con un click sull'apposita “X” accanto al nome del gruppo, esso sarà rimosso dalla casella di input e l'utente non sarà più associato a questo specifico gruppo. Per associare l'utente ai gruppi dell'applicazione basterà fare click sulla casella di input: in questo modo si aprirà un menu a tendina contenente l'elenco fisso di tutti i gruppi dell'applicazione; facendo click su uno di essi questo gruppo verrà introdotto nella casella e lo specifico utente verrà associato ad esso.

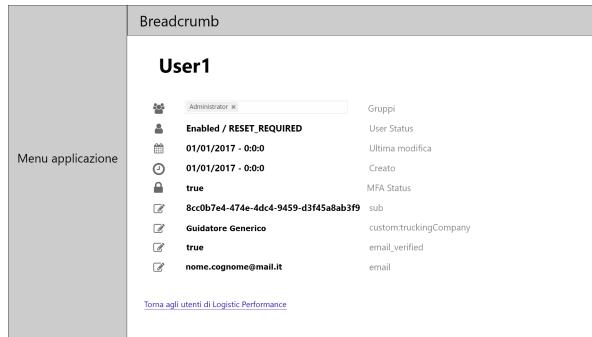


Figura 4.11: Mockup di *Pagina di dettaglio di un singolo utente*

Mockup della User Interface

Essendo l'inglese la lingua dell'applicazione Logistic Performance, tutte le scritte sono state tradotte dall'italiano all'inglese. La casella di input per la selezione multipla (così come le icone e lo stile dei font) sono stati scelti coerentemente con quelli messi a disposizione dal tema grafico Inspinia.

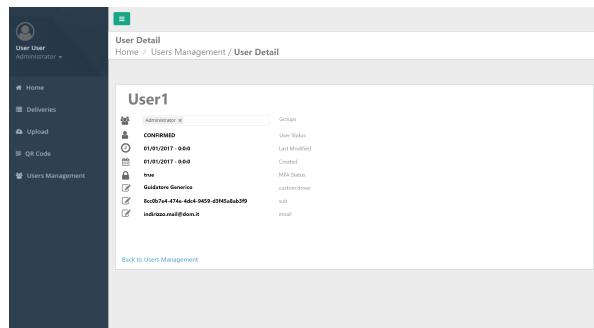


Figura 4.12: Mockup della User Interface di *Pagina di dettaglio di un singolo utente*

4.1.4 Pagina di creazione di un nuovo utente

Wireframe

L'area “Form con campi dati di input” occuperà per intero la pagina web in quanto la *Pagina di creazione di un nuovo utente* conterrà una serie di caselle di input per l'inserimento dei dati per la creazione di un nuovo utente che saranno raggruppati all'interno di un'unica form.



Figura 4.13: Wireframe di *Pagina di creazione di un nuovo utente*

Wireframe con dettaglio delle aree

Essendo possibile creare direttamente un nuovo utente tramite la console AWS usando il servizio Amazon Cognito, è stato scelto di progettare l'inserimento delle proprietà dal front-end per la creazione di un nuovo utente in maniera molto simile a quanto già esistente, ed è per questo che è stato scelto di utilizzare cinque campi di input per l'inserimento di dati da parte dell'utente. Questi campi dati saranno incolonnati e distanziati l'uno con l'altro tramite una lunghezza fissata.

Il pulsante per la creazione di un nuovo utente sarà collocato in basso al centro in posizione fissa. È stato inoltre previsto un link di ritorno alla *Pagina della lista degli utenti* in posizione fissa collocato in basso a sinistra.

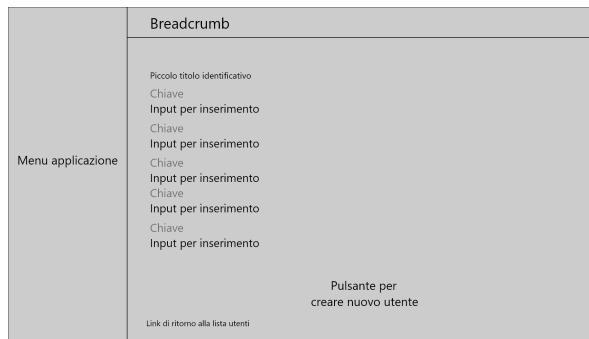


Figura 4.14: Wireframe con dettaglio delle aree di *Pagina di creazione di un nuovo utente*

Mockup

Le cinque proprietà da inserire si dividono in obbligatorie e non obbligatorie. L'unica proprietà obbligatoria è lo username, al quale corrisponderà una casella di input di tipo testuale; il pulsante per la creazione di un nuovo utente non sarà cliccabile fino a quando non sarà stata inserita una qualche stringa nella casella di input che richiede di inserire il nome dell'username.

Si è pensato di progettare la decisione opzionale di ricezione della notifica dell'avvenuta creazione dell'utente tramite una doppia *checkbox*. Si potrà scegliere se il nuovo utente non debba ricevere alcuna notifica della creazione, oppure che debba riceverla tramite Email, SMS o entrambi i servizi (questo è reso possibile grazie a un servizio di Amazon Cognito).

Tutte le caselle di input per l'inserimento delle proprietà in formato stringa conterranno una stringa placeholder (ciascuna riferita all'apposita proprietà) che informerà l'utente sulla tipologia del dato che dovrà inserire.

Qualunque sia lo sbaglio che l'utente possa commettere nell'inserimento delle proprietà all'interno delle caselle di input (username già esistente, lunghezza della password errata, formato email non valido ecc...), questo farà comparire un messaggio di errore posto subito al di sopra del pulsante per la creazione di un nuovo utente che informerà l'utente della tipologia dell'errore riscontrato.

Figura 4.15: Mockup di *Pagina di creazione di un nuovo utente*

Mockup della User Interface

Essendo l'inglese la lingua dell'applicazione Logistic Performance, tutte le scritte sono state tradotte dall'italiano all'inglese. Lo stile delle checkbox è stato mantenuto come quello classico messo a disposizione direttamente da HTML (senza avvalersi del tema grafico Inspinia). Le lunghezze di tutti i campi di input testuali sono state invece progettate per riempire gran parte della pagina web e sono state scelte coerentemente con i campi di input messi a disposizione del tema grafico Inspinia.

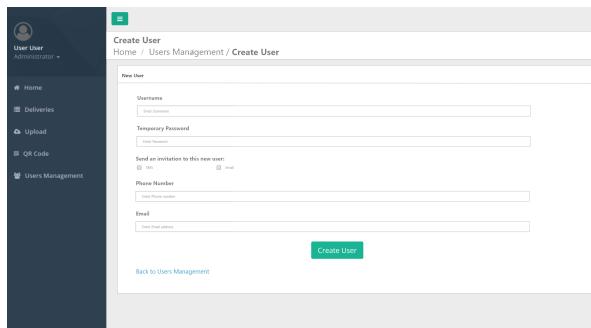


Figura 4.16: Mockup della User Interface di *Pagina di creazione di un nuovo utente*

4.2 Progettazione architetturale

Per progettare l'architettura del front-end dell'applicazione è stata utilizzata una strategia top-down. Il sistema viene descritto partendo da una visione generale (ad alto livello), per poi scendere nel particolare delle parti individuate. Nella visione ad alto livello si può vedere che il sistema è composto da diverse parti che interagiscono tra loro. Chiameremo gli aggregati logici contententi classi, template e fogli di stile **componenti** (in italiano) e non **components** per disambiguare da ciò che sono le componenti di Angular 4: ci si riferirà infatti sempre con la parola *component* (in inglese), alla specifica classe associata a una vista grafica dell'applicazione (si veda l'appendice B su Angular per la definizione di *component*).

In questa prima parte di progettazione ci si concentrerà sulle componenti e sulle classi che le compongono. Ogni componente e ogni classe avrà un unico ruolo ben definito (coerentemente con il *Single Responsibility Principle*). Per modellare la logica di interazione tra le componenti saranno utilizzati diagrammi dei package (un package nel diagramma corrisponderà a un componente) e delle classi conformi al formalismo di *UML 2.0*.

Ci sarà una descrizione soltanto delle componenti delle quali è stata effettuata la progettazione: nei diagrammi queste componenti sono state colorate in arancione. Le componenti colorate in bianco erano già presenti nell'applicazione e non riguardano il progetto di stage; ciononostante sono state inserite comunque per rimarcare le dipendenze con le altre componenti, ma non verranno approfondite nel dettaglio in questo documento.

- Le classi in verde si riferiscono ai *components* di Angular: essi rappresentano sezioni vere e proprie delle pagine web e queste classi sono implicitamente associate a un template HTML e a un foglio di stile CSS (questi file non saranno mai esaminati nel documento ma sono comunque stati fondamentali per rendere possibili alcune cose che il codice di business non avrebbe potuto fare).

- Le classi in giallo rappresentano i *services* di Angular.

Le dipendenze tra le classi e i *services* di Angular sono risolte sempre in maniera efficiente tramite una implementazione del *design pattern* Dependency Injection.

- Le classi in azzurro sono delle classi già esistenti messe a disposizione dal framework Angular o dall'SDK JavaScript. Esse non saranno mai descritte nel dettaglio (inoltre può capitare che tra di esse ci siano dei particolari *services* e anche in questo caso essi appariranno sempre in azzurro e non in giallo).

app

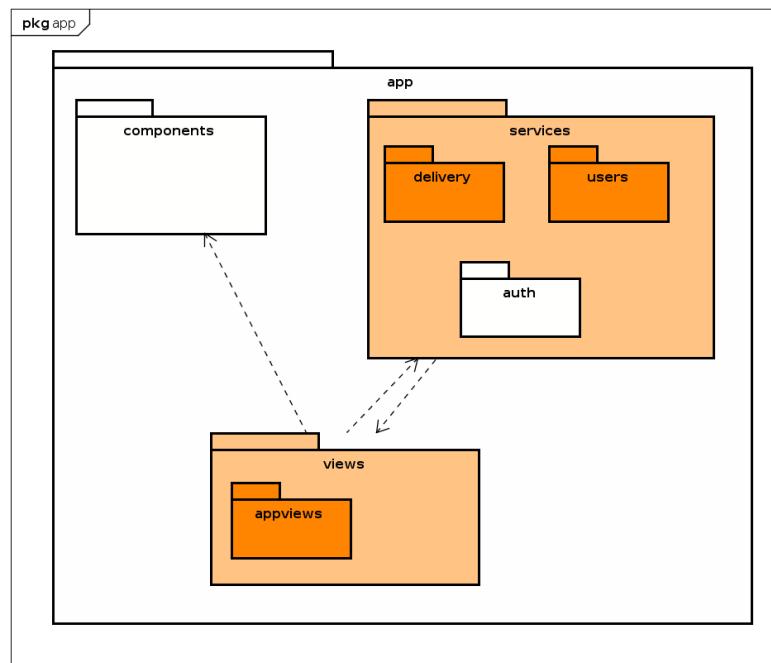


Figura 4.17: app

* Descrizione

Componente principale dell'applicazione.

* Funzionamento

La componente **components** faceva già parte dell'applicazione e contiene un insieme di classi che si occupano del comportamento di alcuni *components* generali di tutta l'applicazione (essi riguardano per esempio il footer o il menu di navigazione).

La componente **views** si riferisce a tutte le classi *component* associate all'applicazione vera e propria. Infine, la componente **services** contiene tutte le classi *services* dell'applicazione.

* **Componenti contenute:**

- **components**
- **services**
- **views**

delivery

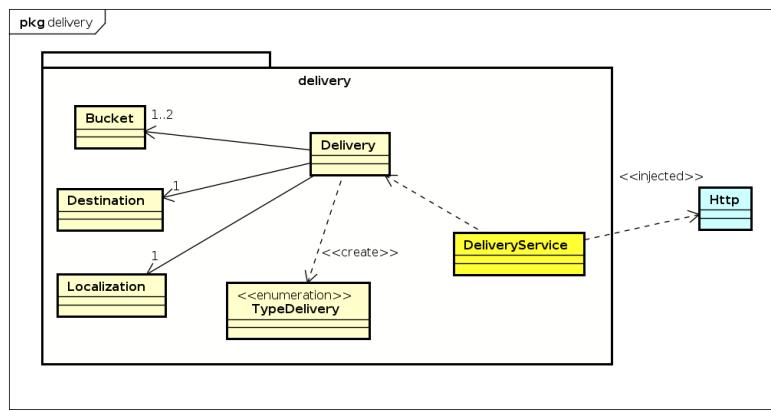


Figura 4.18: delivery

* **Descrizione**

Componente contenente un insieme di classi il cui scopo è modellare i dati delle consegne.

* **Funzionamento**

La classe **Delivery** è la classe che nell'applicazione rappresenta una consegna vera e propria (i suoi campi dati sono definiti come proprietà della consegna). Poichè una generica consegna contiene al proprio interno anche dei dati strutturati, essi vengono definiti come oggetti **Bucket**, **Destination** e **Localization** (la classe **Delivery** avrà una dipendenza verso queste classi in quanto avrà dei campi dati di queste tipologie). Questa classe avrà anche un apposito metodo che restituirà una istanza del tipo enumerazione **TypeDelivery**, che può assumere quattro diversi valori che rappresentano le quattro diverse tipologie della consegna. Il *service* di Angular **DeliveryService** è stato progettato con lo scopo di creare dei metodi che permettessero di manipolare i dati delle consegne attraverso una interazione con il back-end dell'applicazione tramite Amazon API Gateway. Per fare ciò questa classe si serve del *service* **Http**: servizio offerto da Angular per la

gestione delle chiamate HTTP².

users

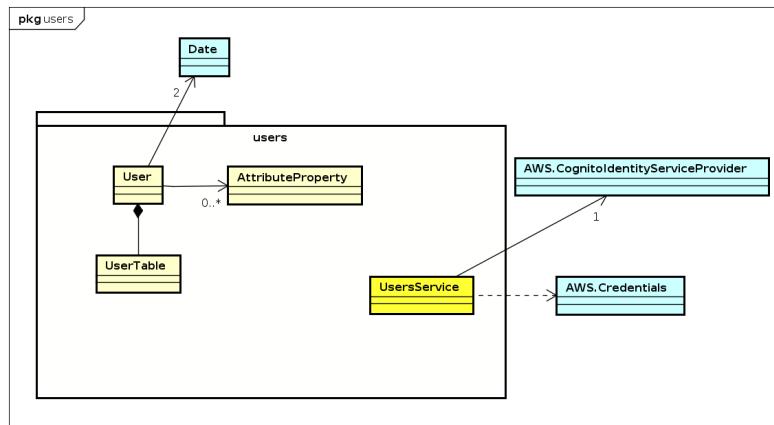


Figura 4.19: users

* Descrizione

Componente contenente un insieme di classi il cui scopo è modellare i dati degli utenti dell'applicazione.

* Funzionamento

La classe **User** è la classe che nell'applicazione rappresenta un generico utente dello User Pool di Amazon Cognito. Essa ha un insieme di campi dati che si riferiscono alle proprietà di questo utente, tra cui le due date di creazione e modifica (ecco perché tale classe ha una dipendenza con esattamente due istanze della classe **Date** di JavaScript/TypeScript) e un aggregato di oggetti di tipo **AttributeProperty** (classe che si riferisce a una generica proprietà di un utente). La classe **UserTable** rappresenta un sottoinsieme di proprietà della classe **User** (nello specifico contiene come campi dati solo ed esclusivamente le proprietà di un utente che dovranno essere visualizzati nella tabella degli utenti) e dunque, pur non essendo legata ad essa da campi dati o invocazioni di metodi, ne è comunque dipendente.

Il *service* **UserService** è utilizzato per il recupero e l'aggiornamento dei dati riguardanti gli utenti dell'applicazione attraverso interazioni con il back-end. Queste interazioni avvengono solo ed esclusivamente tramite metodi messi a disposizione dall'SDK JavaScript per Amazon Cognito (come spiegato nel Paragrafo 1.2.3): da qui la dipendenza verso le classi dell'SDK **AWS.CognitoIdentityServiceProvider** (classe che mette a disposizione tutti i metodi per interagire con lo User Pool)

²Nel Paragrafo 4.3 verranno descritte nel dettaglio solo ed esclusivamente le caratteristiche di **DeliveryService** progettate e poi implementate per la realizzazione dei requisiti visti nel Capitolo 2. Tale classe era già esistente e conteneva anche una serie di metodi utili ma per altre parti dell'applicazione che non sono state assegnate come obiettivi dello stage.

e `AWS.Credentials` (classe la cui istanza si riferirà allo specifico User Pool di AWS ottenuto tramite credenziali AWS IAM).

appviews

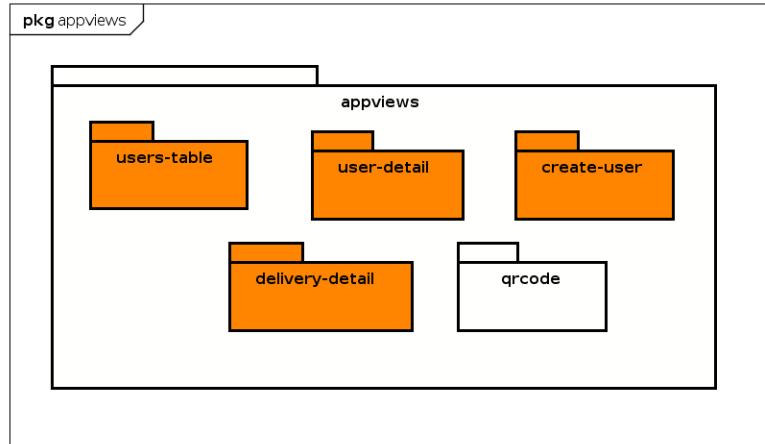


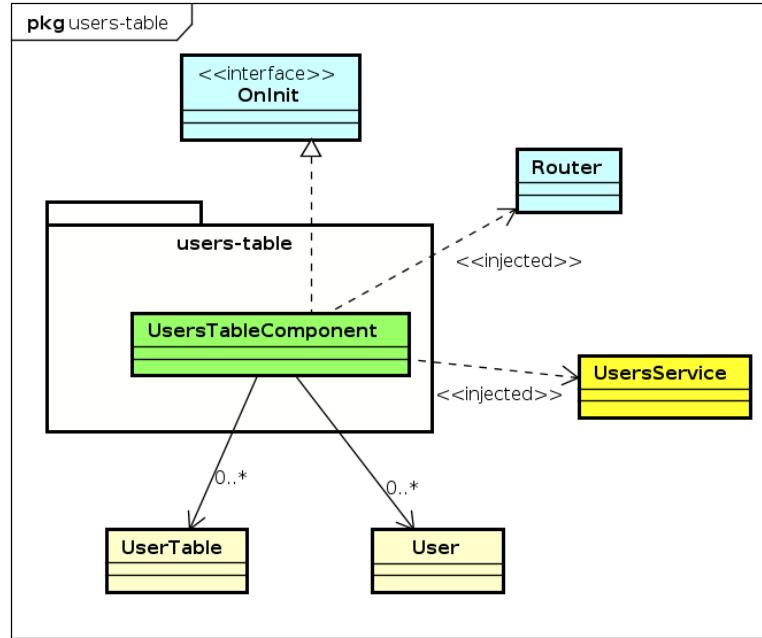
Figura 4.20: appviews

* **Descrizione**

Componente contenente altre componenti logiche: ciascuna di esse riguarda una specifica vista di una intera pagina web.

* **Componenti contenute:**

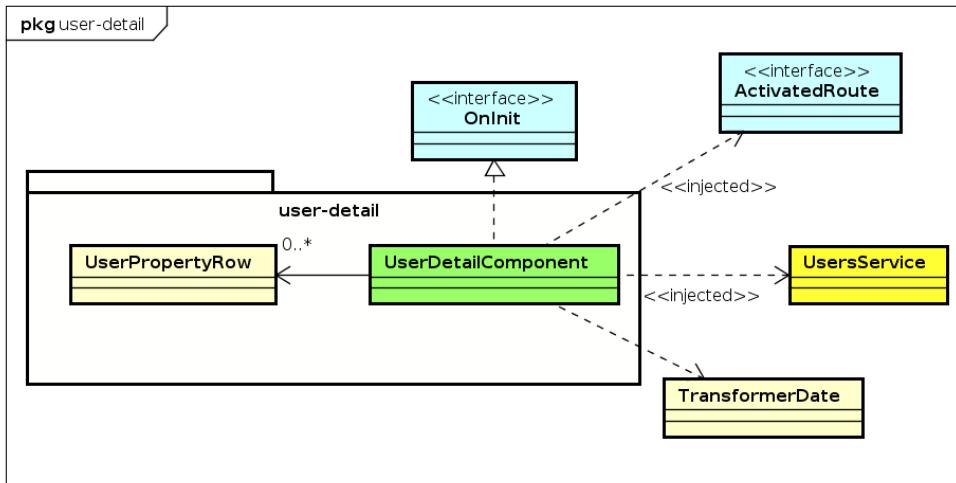
- `users-table`
- `user-detail`
- `create-user`
- `delivery-detail`
- `qrcode`

users-table**Figura 4.21:** users-table*** Descrizione**

Componente che riguarda l'intera pagina web *Pagina della lista degli utenti*.

*** Funzionamento**

La classe *component* di Angular *UsersTableComponent* è usata per riferirsi all'intera pagina web *Pagina della lista degli utenti*. Questa classe avrà relazioni con diverse componenti esterne. Per prima cosa avrà un aggregato di oggetti di tipo *User*; questo rappresenterà l'elenco completo degli utenti dell'applicazione che dovranno essere rappresentati nella tabella. È a partire da questo aggregato che la classe *UsersTableComponent* crea un altro aggregato di oggetti di tipo *UserTable*, che sono oggetti contenenti un sottoinsieme delle proprietà di un singolo utente; nello specifico contengono solamente le proprietà che devono essere rappresentate nella tabella. Queste ultime due classi fanno parte della componente *users*, dentro *services*). *UsersTableComponent* avrà anche una relazione con *Router*, *service* di Angular che rende disponibili diversi servizi per il *routing* (in quanto la tabella dovrà permettere all'utente di andare a vedere il dettaglio di un utente spostandosi nell'apposita *Pagina di dettaglio di un singolo utente* tramite un link, con meccanismo di routing con passaggio di parametri) e con *UserService*, *service* della componente *services* per la gestione dei dati riguardanti gli utenti che permetterà di ottenere tramite una chiamata opportuna al back-end la lista completa di tutti gli utenti dell'applicazione come insieme di oggetti *User*.

user-detail**Figura 4.22:** user-detail*** Descrizione**

Componente che riguarda l'intera pagina web *Pagina di dettaglio di un singolo utente*.

*** Funzionamento**

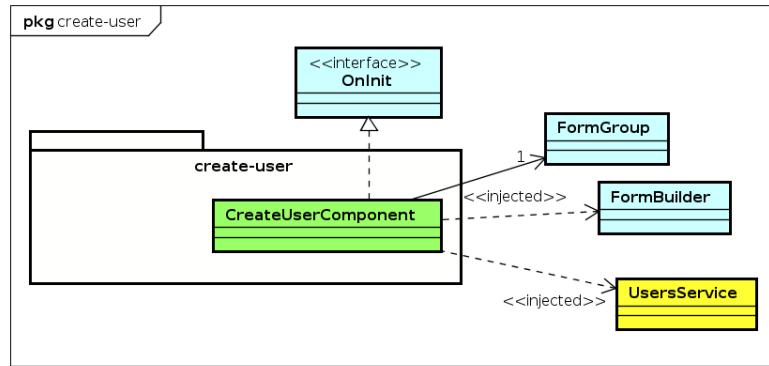
La classe *component* di Angular `UserDetailComponent` è usata per riferirsi all'intera pagina web *Pagina di dettaglio di un singolo utente*. Questa classe ha una relazione con un aggregato di oggetti `UserPropertyRow` (ognuno di questi oggetti rappresenta specifica riga statica dell'intera lista delle proprietà), e con `TransformerDate`, classe della componente `services`, per garantire una corretta visualizzazione a schermo degli orari di creazione e modifica degli utenti. Ha anche una relazione con l'interfaccia `ActivatedRoute`, interfaccia di Angular che rende disponibili diversi servizi per il *routing* (in quanto la pagina web recupererà l'informazione contenente l'identificativo dello specifico utente a partire da un'altra pagina web, tramite il meccanismo di routing con passaggio di parametri) e `UsersService`, service della componente `services` che permetterà di associare i gruppi allo specifico utente (o rimuoverli) dinamicamente direttamente dal front-end.

create-user*** Descrizione**

Componente che riguarda l'intera pagina web *Pagina di creazione di un nuovo utente*.

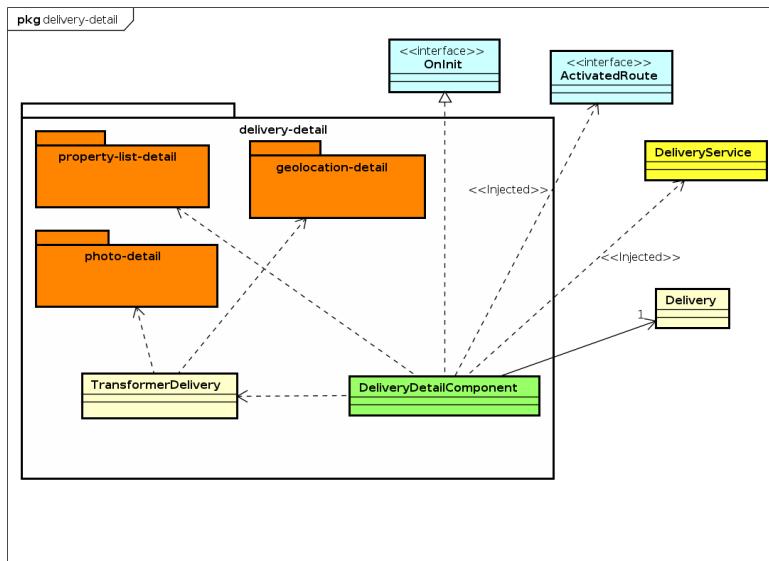
*** Funzionamento**

La classe *component* di Angular `CreateUserComponent` è usata per riferirsi all'intera pagina web *Pagina di creazione di un nuovo utente*. Basandosi la pagina su un'unica form contenente diversi campi di input, questa classe ha una

**Figura 4.23:** create-user

relazione con un oggetto di tipo `FormGroup` (classe di Angular che rappresenta appunto un gruppo di campi di input, e implementa il meccanismo delle *reactive forms*) e una relazione con il *service* di Angular `FormBuilder` (quest'ultimo serve soltanto per facilitare la creazione di tutti i campi dati di input di una singola form). La classe ha anche una relazione con `UsersService`, *service* della componente `services` che permetterà la creazione di un nuovo utente direttamente dal front-end.

delivery-detail

**Figura 4.24:** delivery-detail

* Descrizione

Componente che riguarda l'intera pagina web *Pagina di dettaglio di una consegna*.

* **Funzionamento**

La classe *component* di Angular *DeliveryDetailComponent* è usata per riferirsi all'intera pagina web *Pagina di dettaglio di una consegna*. Questa classe avrà una relazione con un oggetto *Delivery* (della componente *delivery*, dentro *services*) che rappresenterà l'unica specifica consegna su cui si basa l'intera pagina. Il suo scopo è quello di smistare i giusti dati a tre *subcomponents* innestate (queste ultime si trovano all'interno delle componenti del sistema *property-list-detail*, *photo-detail* e *geolocation-detail* e si riferiscono alle tre sottosezioni della pagina di dettaglio di una consegna, ovvero la lista di proprietà, la localizzazione e la foto della bolla con orario. Per lo smistamento di questi dati si servirà della classe *TransformerDelivery*, il cui scopo è trasformare un qualsiasi oggetto di tipo *Delivery* passato in input in oggetti nuovi che incapsuleranno soltanto un sottoinsieme delle proprietà di questo oggetto. Questi oggetti avranno tipi diversi e verranno quindi passati in input alla giusta *subcomponent* di *DeliveryDetailComponent*, che in questo modo non dovrà essere responsabile del filtraggio di dati inutili ma invece riceverà un unico oggetto contenente solo e soltanto i dati che le serviranno. *DeliveryDetailComponent* ha anche una relazione con l'interfaccia *ActivatedRoute*, interfaccia di Angular che rende disponibili diversi servizi per il *routing* (in quanto la pagina web recupererà l'informazione contenente l'identificativo della specifica consegna a partire da un'altra pagina web, tramite il meccanismo di routing con passaggio di parametri) e *DeliveryService*, servizio della componente *services* che fornirà l'effettivo oggetto di tipo *Delivery* grazie ad una chiamata HTTP al back-end.

* **Componenti contenute:**

- *property-list-detail*
- *geolocation-detail*
- *photo-detail*

property-list-detail

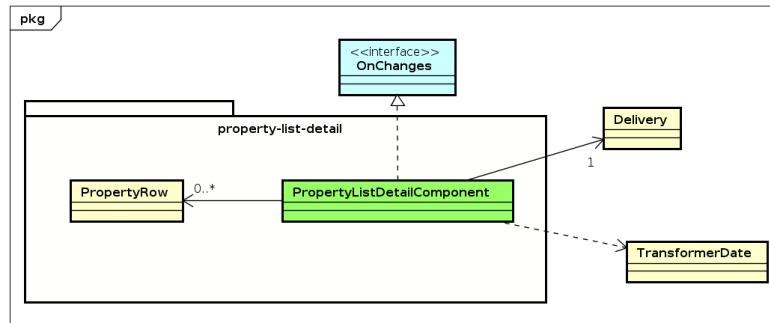


Figura 4.25: property-list-detail

* **Descrizione**

Componente associata alla lista di proprietà di una consegna della *Pagina di dettaglio di una consegna*.

* **Funzionamento** `PropertyListDetailComponent` è la classe *component* di Angular utilizzata per gestire la lista delle proprietà della consegna. Lo scopo di questa classe è quello di gestire la logica applicativa della lista di proprietà (come visualizzare nel giusto formato le informazioni giuste e nella giusta posizione a seconda della tipologia di consegna). A questa classe sarà passato in input (dal template del *component* `DeliveryDetailComponent`) un intero oggetto `Delivery` (della componente `delivery`, dentro `services`) che sarà usato per accedere a quasi tutte le sue proprietà³.

Ha anche una relazione con `TransformerDate`, classe della componente `services` (per garantire una corretta visualizzazione a schermo degli orari), con un aggregato di oggetti di tipo `PropertyRow` e con `DeliveryService`, servizio della componente `services` che permetterà di implementare la funzionalità di modifica di una consegna. La classe `PropertyRow` ha lo scopo istanziare degli oggetti corrispondenti a delle righe da visualizzare nella lista di proprietà. La classe *component* `PropertyListDetailComponent` contiene un aggregato di questi oggetti e ogni oggetto rappresenta proprio una riga dell'elenco delle proprietà. Si sarebbe potuto scrivere tutto ciò direttamente nel template HTML del *component* ma così facendo è stato ottenuto un guadagno in termini di modularità del codice (sarà sempre possibile aggiungere dati dinamicamente per aumentare la lista delle proprietà senza modificare il template HTML).

geolocation-detail

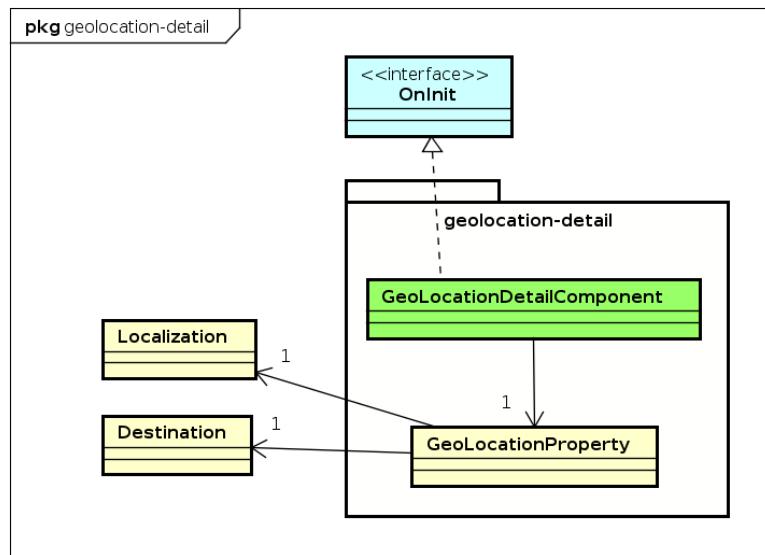


Figura 4.26: geolocation-detail

³Si è scelto di non utilizzare la classe `TransformerDelivery` per il filtraggio dei dati necessari a questo *subcomponent* in quanto la lista delle proprietà necessita di quasi tutti i dati di un singolo oggetto di tipo `Delivery` (al contrario delle altre due *subcomponents*) e quindi è sembrato opportuno passare in input l'intero oggetto.

* **Descrizione**

Componente associata alla parte di localizzazione (contenente l'applicazione esterna Google Maps e l'indirizzo dell'avvenuta consegna) della *Pagina di dettaglio di una consegna*.

* **Funzionamento**

GeolocationDetailComponent è la classe *component* di Angular usata per gestire la logica applicativa della applicazione esterna Google Maps e della visualizzazione dell'indirizzo dell'avvenuta consegna (gestendo i casi in cui essi debbano essere rimpiazzati con dei placeholder). Ha una relazione con **GeoLocationProperty** (classe contenente un sottoinsieme dei campi dati di un oggetto di tipo **Delivery**, precisamente **Localization** e **Destination**, anche loro classi contenute nella componente **delivery** che rappresentano proprio le proprietà utili per la rappresentazione scelta della parte di localizzazione) poichè gli viene passata in input (direttamente dal template del *component DeliveryDetailComponent*) proprio una istanza di questa classe grazie al *transformer*.

photo-detail

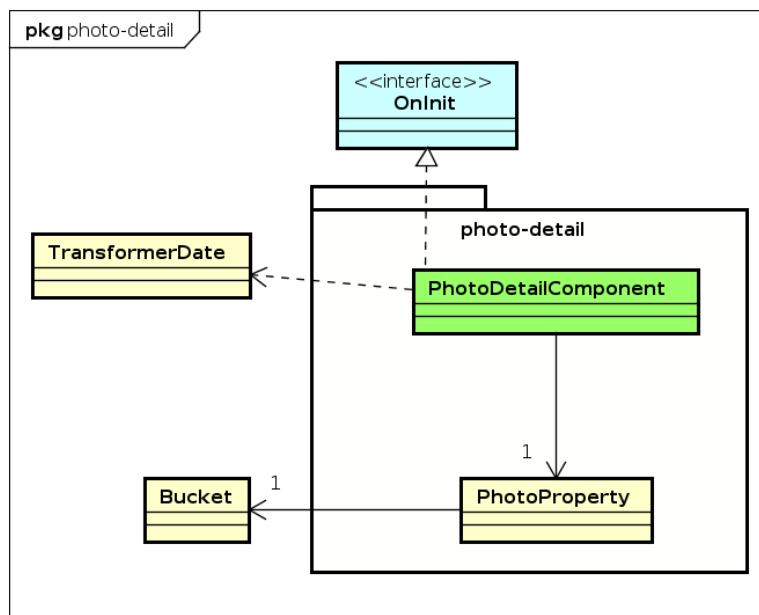


Figura 4.27: photo-detail

* **Descrizione**

Componente associata alla parte della foto della bolla di una avvenuta consegna della *Pagina di dettaglio di una consegna*.

* **Funzionamento**

PhotoDetailComponent è la classe *component* di Angular usata per gestire la logica applicativa della visualizzazione della foto della bolla di consegna e della visualizzazione dell'orario dell'avvenuta consegna (gestendo i casi in cui

essi debbano essere rimpiazzati con dei placeholder). Ha una relazione con `PhotoProperty` (classe contenente un sottoinsieme dei campi dati di un oggetto di tipo `Delivery`, precisamente un oggetto di tipo `Bucket`, che rappresenta proprio la giusta proprietà utile per la rappresentazione scelta della parte di visualizzazione della foto della bolla di una avvenuta consegna) poichè gli viene passata in input (direttamente dal template del `component DeliveryDetailComponent`) proprio una istanza di questa classe grazie al `transformer`. C'è una dipendenza anche verso `TransformerDate`, classe della componente `services`, per garantire una corretta visualizzazione a schermo dell'orario dell'avvenuta consegna.

4.3 Progettazione di dettaglio e codifica

In questa sezione verrà illustrato nel dettaglio il funzionamento delle classi individuate. A differenza della sezione precedente, che si focalizzava sulle interazioni tra le diverse classi, questa sezione si concentra sulle funzionalità di ogni singola classe indipendentemente dal resto dell'architettura. Per ogni classe è presente una breve descrizione della stessa, un elenco dei suoi campi dati e un elenco dei suoi metodi.

DeliveryDetailComponent

DeliveryDetailComponent
<ul style="list-style-type: none"> - delivery : Delivery - route : ActivatedRoute - deliveryService : DeliveryService
<ul style="list-style-type: none"> - getCurrentDelivery() : void + getGeolocationProperty() : GeolocationProperty + getPhotoProperty() : PhotoProperty

Figura 4.28: DeliveryDetailComponent

Descrizione: Classe `component` che rappresenta l'intera *Pagina di dettaglio di una consegna*.

Campi dati:

1. - `delivery`: `Delivery`
Rappresenta la specifica consegna oggetto della pagina di dettaglio della consegna.
2. - `route` : `ActivatedRoute`
Serve per recuperare una informazione contenente l'identificativo della specifica consegna.
3. - `deliveryService` : `DeliveryService`
Fornisce un servizio per recuperare un oggetto `Delivery`.

Metodi:

1. - `getCurrentDelivery()` : `void`
Inizializza il campo dati `delivery` con un oggetto ottenuto a partire dal meccanismo di routing con passaggio di parametri.

2. + `getGeolocationProperty() : GeolocationProperty`

Servendosi della classe `TransformerDelivery` restituisce un oggetto contenente un sottoinsieme delle proprietà dell'oggetto `delivery`, che sarà passato a `GeolocationDetailComponent`.

3. + `getPhotoProperty() : PhotoProperty`

Servendosi della classe `TransformerDelivery` restituisce un oggetto contenente un sottoinsieme delle proprietà dell'oggetto `delivery`, che sarà passato a `PhotoDetailComponent`.

TransformerDelivery

TransformerDelivery
+ <code>getGeolocationFromDelivery(delivery : Delivery) : GeolocationProperty</code>
+ <code>getPhotoFromDelivery(delivery : Delivery) : PhotoProperty</code>

Figura 4.29: TransformerDelivery

Descrizione: Classe con lo scopo di estrarre un certo sottoinsieme di proprietà da un oggetto di tipo `Delivery` e di restituire tale sottoinsieme come un oggetto. Tutti i metodi di questa classe sono statici.

Metodi:

1. + `getGeolocationFromDelivery(delivery : Delivery) : GeolocationProperty`
Costruisce e restituisce un oggetto di tipo `GeolocationProperty` contenente un sottoinsieme delle proprietà dell'oggetto `delivery` passato in input.

2. + `getPhotoFromDelivery(delivery : Delivery) : PhotoProperty`
Costruisce e restituisce un oggetto di tipo `PhotoProperty` contenente un sottoinsieme delle proprietà dell'oggetto `delivery` passato in input.

PropertyListDetailComponent

Descrizione: Classe *component* che rappresenta l'intera lista di proprietà della *Pagina di dettaglio di una consegna*.

PropertyListDetailComponent
- <code>delivery : Delivery</code> - <code>propertyRows : PropertyRow[]</code> - <code>deliveryService : DeliveryService</code>
- <code>getViewFormatDate() : string</code> - <code>putNewField(newField : string, parameter : string) : void</code>

Figura 4.30: PropertyListDetailComponent

Campi dati:

1. - `delivery:Delivery`

Rappresenta la consegna dalla quale estrapolare i dati da visualizzare nell'elenco delle proprietà.

2. - `propertyRows : PropertyRow[]`

Array di oggetti di tipo `PropertyRow`: questo si riferisce all'elenco grafico delle proprietà e ogni oggetto è associato a una particolare riga della lista.

3. - `deliveryService : DeliveryService`

Fornisce un servizio per modificare un oggetto `Delivery`.

Metodi:

1. - `getViewFormatDate() : string`

Restituisce (tramite la classe `TransformerDate`) una stringa della data stimata di consegna nel formato “GG:MM:YYYY - H:M:S”.

2. - `putNewField(newField : string, parameter : string) : void`

Utilizzando l'istanza `deliveryService`, questo metodo modifica il valore di una particolare proprietà della specifica consegna (questa proprietà è identificata dalla stringa `parameter`), sostituendo tale valore con la stringa `newField`.

PropertyRow

PropertyRow	
- <code>parameter : string</code>	
- <code>form : FormControl</code>	
- <code>isEditable : boolean</code>	
- <code>isEditClick : boolean = false</code>	
- <code>icon : string</code>	
- <code>fieldKey : string</code>	
+ <code>generateForm() : void</code>	

Figura 4.31: PropertyRow

Descrizione: Classe che rappresenta una specifica riga associata a una specifica proprietà di una consegna. Tale riga fa parte di un insieme di righe che forma la lista di proprietà di una consegna.

Campi dati:

1. - `parameter : string`

Stringa rappresentante un effettivo campo dati di un generico oggetto `Delivery` a cui si riferisce la riga della lista di proprietà (tale stringa sarà identica al nome di un campo dati di un oggetto `Delivery`).

2. - `form: FormControl`

Campo dati di tipo `FormControl` (classe di Angular per implementare le *reactive forms*) il cui scopo è fare apparire una casella di input testuale (con valore da inserire obbligatorio).

3. - `isEditable : boolean`

Boleano che vale `true` se il dato oggetto della riga è modificabile da un amministratore, `false` altrimenti.

4. - `isEditClick : boolean`

Boleano che vale `true` se l'amministratore ha scelto di modificare il dato della riga (in tal caso su schermo appare la casella di input), `false` altrimenti.

5. - `icon : string`

Stringa contenente l'identificativo dell'icona (del tema Inspinia).

6. - `fieldKey : string`

Stringa contenente la chiave del dato oggetto della riga (ovvero la sua descrizione statica).

Metodi:

1. + `generateForm(): void`

Metodo che visualizza e nasconde la casella di input per la modifica al click dell'amministratore sull'icona della modifica, settando lo stato dei campi dati `isEditable` e `isEditClick`.

GeolocationDetailComponent

GeolocationDetailComponent
- <code>geolocationProperty : GeolocationProperty</code>
+ <code>isValidGeo(): boolean</code>
+ <code>getGeoLocationAddress(): string</code>

Figura 4.32: GeolocationDetailComponent

Descrizione: Classe *component* che rappresenta la parte della *Pagina di dettaglio di una consegna* dedicata alla localizzazione (ovvero geolocalizzazione e visualizzazione dell'indirizzo di consegna).

Campi dati:

1. - `geolocationProperty : GeolocationProperty`

Oggetto passato in input contenente i dati di una consegna che devono essere utilizzati per la rappresentazione della localizzazione.

Metodi:

1. + `isValidGeo(): boolean`

Metodo che restituisce `true` se le proprietà dell'oggetto `geolocationProperty` sono diverse da `null`, `false` altrimenti.

2. + `getGeoLocationAddress() : string`

Metodo che restituisce una stringa che contiene l'indirizzo dell'avvenuta consegna se questo esiste, o la stringa “Localization not available” altrimenti.

GeolocationProperty

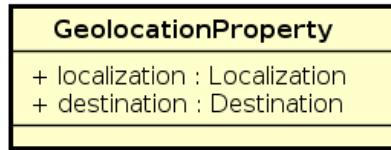


Figura 4.33: GeolocationProperty

Descrizione: Classe contenente un sottoinsieme di campi dati di un oggetto Delivery. Tale sottoinsieme riguarda i dati della localizzazione di una consegna.

Campi dati:

1. + localization: Localization
Oggetto utilizzato per recuperare le coordinate del luogo in cui è stata effettuata la consegna.
2. + destination: Destination
Oggetto utilizzato per recuperare l'indirizzo dell'avvenuta consegna.

PhotoDetailComponent

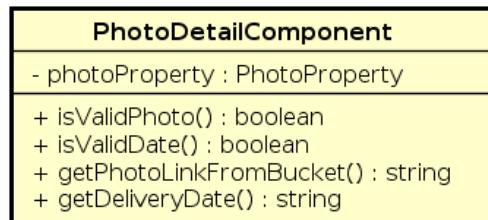


Figura 4.34: PhotoDetailComponent

Descrizione: Classe *component* che rappresenta la parte della *Pagina di dettaglio di una consegna* dedicata alla visualizzazione della bolla di consegna (e anche alla visualizzazione dell'esatto orario dell'avvenuta consegna).

Campi dati:

1. - photoProperty : PhotoProperty
Oggetto passato in input contenente i dati di una consegna che devono essere utilizzati per la rappresentazione della foto della bolla di consegna e dell'orario di avvenuta consegna.

Metodi:

1. + isValidPhoto() : boolean
Metodo che restituisce `true` se nell'oggetto `photoProperty` il link del `bucket S3` in cui è localizzata l'immagine della foto della bolla è diverso da `null`, `false` altrimenti.

2. + `isValidDate(): boolean`

Metodo che restituisce `true` se nell'oggetto `photoProperty` la stringa della data dell'avvenuta consegna è diversa da `null`, `false` altrimenti.

3. + `getPhotoLinkFromBucket(): string`

Metodo che restituisce una stringa contenente il link del *bucket S3* in cui è localizzata l'immagine della foto della bolla dell'avvenuta consegna.

4. + `getDeliveryDate(): string`

Metodo che restituisce una stringa che contiene la data dell'avvenuta consegna se questa esiste (tramite la classe `TransformerDate`, nel formato “GG:MM:YYYY - H:M:S”), o la stringa “Delivery not sent” altrimenti.

PhotoProperty

PhotoProperty
+ <code>photoLink : Bucket</code>
+ <code>deliveryDate() : string</code>

Figura 4.35: PhotoProperty

Descrizione: Classe contenente un sottoinsieme di campi dati di un oggetto *Delivery*. Tale sottoinsieme riguarda i dati utili per la visualizzazione della foto della bolla e dell'orario di una consegna.

Campi dati:1. + `photoLink: Bucket`

Oggetto utilizzato per recuperare il link del *bucket S3* in cui è localizzata l'immagine della foto della bolla di una avvenuta consegna.

2. + `deliveryDate: string`

Stringa utilizzata come orario (giorno e ora) dell'avvenuta consegna.

Bucket

Bucket
+ <code>bucketName : string</code>
+ <code>keyName : string</code>
+ <code>uploadPreSignedUrl : string</code>
- <code>downloadPreSignedUrl : string</code>
+ <code>getDownloadUrl(): string</code>

Figura 4.36: Bucket

Descrizione: Lo scopo di questa classe è rappresentare un oggetto (dato non strutturato come una immagine) caricato nello *storage* di Amazon S3 accessibile tramite un

URL.

Campi dati:

1. + `bucketName` : string
Stringa che si riferisce al nome del bucket di Amazon S3 dove è collocata la risorsa.
2. + `keyName` : string
Stringa che si riferisce a un identificativo univoco all'interno del bucket Amazon S3.
3. + `uploadPreSignedUrl`: string
Stringa che si riferisce a un URL utilizzato per l'upload di un file.
4. - `downloadPreSignedUrl`: string
Stringa che si riferisce a un URL utilizzato per il download di un file (questa stringa potrebbe non essere disponibile per alcuni oggetti Bucket).

Metodi:

1. + `getDownloadUrl()` : string
Restituisce una stringa contenente il link di download (ovvero il campo `downloadPreSignedUrl`) se questo esiste, altrimenti restituisce `null`.

Destination

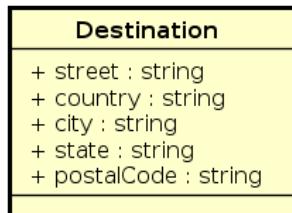


Figura 4.37: Destination

Descrizione: Questa classe rappresenta un qualsiasi indirizzo fisico del mondo reale.

Campi dati:

1. + `street` : string
Stringa che si riferisce al nome di una strada.
2. + `country` : string
Stringa che si riferisce al nome di un paese.
3. + `city`: string
Stringa che si riferisce al nome di una città.
4. + `state`: string
Stringa che si riferisce al nome di uno stato.
5. + `postalCode`: string
Stringa che si riferisce a un codice postale.

Localization

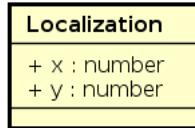


Figura 4.38: Localization

Descrizione: Questa classe rappresenta le coordinate GPS di una qualsiasi posizione nel mondo reale a partire dalle due coordinate x e y.

Campi dati:

1. + x : number
Coordinata GPS x (longitudine).
2. + y : number
Coordinata GPS y (latitudine).

Delivery



Figura 4.39: Delivery

Descrizione: Classe che rappresenta una generica consegna all'interno dell'applicazione.

Campi dati:

1. + deliveryId : string
Stringa contenente l'identificativo univoco della consegna.
2. + deliveryNumber: string
Stringa che si riferisce al numero identificativo della consegna.

3. + `deliveryDate: string`
Stringa che si riferisce all'effettivo orario dell'avvenuta consegna.
4. + `estimatedDeliveryDate : string`
Stringa che si riferisce all'orario stimato di una consegna.
5. + `destination: Destination`
Oggetto di tipo `Destination` che si riferisce all'indirizzo al quale deve essere effettuata la consegna.
6. + `photoLink: Bucket`
Oggetto di tipo `Bucket` che si riferisce alla posizione nel bucket di Amazon S3 all'interno della quale è caricata la fotografia della bolla di consegna.
7. + `disputedPhotoLink: Bucket`
Oggetto di tipo `Bucket` che si riferisce alla posizione nel bucket di Amazon S3 all'interno della quale è caricata una fotografia del dettaglio di una contestazione nel caso in cui la consegna sia con riserva.
8. + `geoLocation: Localization`
Oggetto di tipo `Localization` che si riferisce alle coordinate GPS del punto esatto in cui è stata scattata la fotografia della bolla di consegna (che è il punto in cui tale consegna è stata effettuata).
9. + `isDisputed: boolean`
Campo dati booleano che vale `true` se la consegna è stata contestata (ovvero è etichettata come consegna con riserva), `false` altrimenti.
10. + `driver: string`
Stringa che identifica il nome del trasportatore della consegna.

Metodi:

1. - `isDelay(): boolean`
Metodo che restituisce un booleano che vale `true` se la consegna è stata consegnata con ritardo, `false` altrimenti. Si basa sostanzialmente sul calcolo della differenza tra data stimata e data effettiva della consegna.
2. - `isContested(): boolean`
Metodo che restituisce un booleano che vale `true` se la consegna è stata contestata, `false` altrimenti.
3. - `isOk(): boolean`
Metodo che restituisce un booleano che vale `true` se la consegna è stata consegnata con successo, `false` altrimenti.
4. - `isNotDelivered(): boolean`
Metodo che restituisce un booleano che vale `true` se la consegna non è ancora stata consegnata, `false` altrimenti.
5. + `getDeliveryStatus(): TypeDelivery`
Metodo che restituisce una enumerazione `TypeDelivery` che indicherà la specifica tipologia della consegna tra le quattro esistenti.

DeliveryService

DeliveryService
- deliveriesUrl : string
+ getDeliveryById(deliveryId : string) : Observable<Delivery>
+ putDelivery(delivery : Delivery) : Observable<any>

Figura 4.40: DeliveryService

Descrizione: Classe *service* di Angular per la gestione delle chiamate HTTP al servizio del back-end Amazon API Gateway con lo scopo di accedere in lettura e in scrittura ai dati riguardanti le consegne. Non verranno descritti tutti i suoi metodi ma solo quelli progettati per soddisfare i requisiti assegnati.

Campi dati:

1. - **deliveriesUrl : string**

Stringa contenente l'URL all'API contenente i metodi per l'interazione con i dati delle consegne esposta da Amazon API Gateway.

Metodi:

1. + **getDeliveryById(deliveryId : string) : Observable<Delivery>**

Metodo che restituisce dal back-end (tramite una chiamata HTTP GET) un oggetto di tipo **Delivery** a partire da una stringa **deliveryId** riferita al suo identificativo univoco. L'oggetto **Delivery** viene restituito attraverso un oggetto **Observable** (che in Angular è un flusso di dati asincrono gestibile tramite operatori specializzati come se fosse un tipo contenitore).

2. + **putDelivery(delivery : Delivery) : Observable<any>**

Metodo che riceve un parametro di tipo **Delivery** e tramite una chiamata HTTP PUT al back-end, aggiorna i dati di quella specifica consegna in base alle modifiche effettuate nel front-end da un utente amministratore.

UsersTableComponent

UsersTableComponent
- users : User[] - usersFiltered : UserTable[] - rows : Array<any> = [] - columns : Array<any> - page : number = 1 - itemsPerPage : number = 8
- onChangeTable(config : any, page : any) : any - changePage(page : any, data : Array<any>) : Array<any> + onCellClick(data : any) : any

Figura 4.41: UsersTableComponent

Descrizione: Classe *component* di Angular per la rappresentazione della pagina web *Pagina della lista degli utenti*, che è una tabella contenente i dati di tutti gli utenti dell'applicazione⁴.

Campi dati:

1. - `users : User[]`
Array di oggetti `User`. Rappresenta tutti gli utenti dell'applicazione.
2. - `usersFiltered : UserTable[]`
Array di oggetti `UserTable`. Rappresenta gli oggetti contenenti le proprietà degli utenti a cui effettivamente farà riferimento la tabella.
3. - `rows : Array<any>`
Array di oggetti che rappresenta le righe della tabella.
4. - `columns : Array<any>`
Array di oggetti che rappresenta le colonne della tabella.
5. - `page : number`
Campo dati che si riferisce al numero di pagine (ovvero facciate) della tabella (inizializzato al valore 1 di default).
6. - `itemsPerPage:number`
Campo dati che si riferisce al numero massimo di righe per ogni facciata della tabella (inizializzato al valore 8 di default).

Metodi:

1. - `onChangeTable(config : any, page : any) : any`
Metodo che viene invocato (dal template) al click dell'utente del pulsante per cambiare facciata della tabella.
2. - `changePage(page : any, data : Array<any>) : Array<any>`
Metodo che viene automaticamente invocato che serve a gestire il cambio di una facciata all'interno della tabella.
3. + `onCellClick(data : any) : any`
Metodo che viene invocato (dal template) ogniqualvolta viene eseguito un click su una riga. Questo metodo sfrutta il meccanismo di *routing* per permettere la navigazione alla pagina successiva (ovvero la pagina di dettaglio dello specifico utente cliccato).

⁴Essendo già stata progettata la tabella degli utenti in modo quasi identico alla già esistente tabella delle consegne presente nella *Pagina della lista delle consegne* (pagina di cui non è stata chiesta la realizzazione durante lo stage perché già esistente), i metodi e i campi dati di questa classe sono stati progettati in modo molto simile a questa componente e non sono presenti tutti.

AttributeProperty

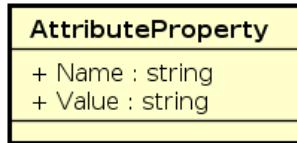


Figura 4.42: AttributeProperty

Descrizione: Classe utilizzata per rappresentare un generico attributo opzionale di un utente in formato *chiave-valore*⁵.

Campi dati:

1. + Name: string
Stringa che indica il nome della proprietà.
2. + Value : boolean
Stringa che indica l'effettivo valore della proprietà.

User

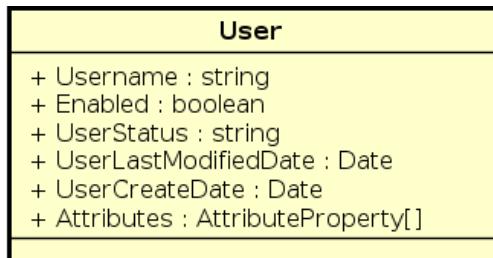


Figura 4.43: User

Descrizione: Classe utilizzata per rappresentare un utente dell'applicazione.

Campi dati:

1. + Username: string
Stringa che indica l'username dell'utente.
2. + Enabled : boolean
Booleano che vale `true` se l'utente è stato abilitato, `false` altrimenti.
3. + UserStatus : string
Stringa che indica lo stato in cui si trova l'utente (ovvero se l'utente ha confermato la sua registrazione o meno).

⁵I nomi dei campi dati iniziano con lettera maiuscola senza rispettare la classica convenzione della programmazione ad oggetti perchè altrimenti non sarebbe stata possibile una associazione effettiva con i dati restituiti dal back-end dell'applicazione, i cui nomi devono matchare perfettamente con i nomi arbitrari dei campi dati.

4. + **UserLastModifiedDate:** Date

Oggetto di tipo **Date** che indica la data dell'ultima volta in cui sono state modificate le proprietà dell'utente.

5. + **UserCreateDate:** Date

Oggetto di tipo **Date** che indica la data di creazione dell'utente.

6. + **Uttributes :** AttributeProperty[]

Array di oggetti che rappresenta l'insieme variabile di tutte le altre proprietà opzionali che un utente può avere.

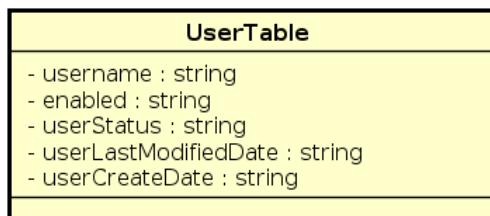
UserTable

Figura 4.44: UserTable

Descrizione: Classe utilizzata per rappresentare un sottoinsieme di proprietà di un oggetto **User**. Questo sottoinsieme di proprietà è uguale all'insieme di proprietà che devono essere visibili nella tabella degli utenti dell'applicazione.

Campi dati:1. - **username:** string

Stringa che indica l'username dell'utente

2. - **enabled :** string

Stringa che indica se l'utente è abilitato o no (assume i valori **true** e **false**).

3. - **userStatus :** string

Stringa che indica lo stato in cui si trova l'utente (ovvero se l'utente ha confermato la sua registrazione o meno).

4. - **userLastModifiedDate :** string

Stringa che indica la data dell'ultima volta in cui sono state modificate le proprietà dell'utente.

5. - **userCreateDate:** string

Stringa che indica la data di creazione dell'utente.

UsersService

UsersService
- cognitoIdentityServiceProvider : AWS.CognitoIdentityServiceProvider
+ getCognitoUsers() : Promise<AWS.CognitoIdentityServiceProvider.ListUsersResponse>
+ createCognitoUsers(username : string, password : string, emailValue : string, phone_numberValue : string, sms : string, email : string) : string
+ listGroup() : Promise<AWS.CognitoIdentityServiceProvider.ListGroupsResponse>
+ listGroupForUser(username : string) : Promise<AWS.CognitoIdentityServiceProvider.AdminListGroupsForUserResponse>
+ getUserProperties(username : string) : Promise<AWS.CognitoIdentityServiceProvider.Admin GetUserResponse>
+ addUserToGroup(groupName : string, username : string) : void
+ removeUserFromGroup(groupName : string, username : string) : void

Figura 4.45: UsersService

Descrizione: Classe *service* di Angular per la gestione dei dati riguardanti gli utenti dell'applicazione. Questi dati vengono recuperati e aggiornati tramite interazioni con il back-end che fanno largamente uso dell'SDK JavaScript per Amazon Cognito, grazie all'uso della classe `AWS.CognitoIdentityServiceProvider`, propria di questo SDK, che espone ogni API di Amazon Cognito tramite comodi metodi di utilità⁶.

Campi dati:

1. - `cognitoIdentityServiceProvider` : `AWS.CognitoIdentityServiceProvider`
Campo dati che si riferisce all'intero servizio Amazon Cognito del back-end dell'applicazione. Tramite questo campo dati è possibile l'invocazione di tutti i metodi di utilità per l'interazione con il back-end tramite l'SDK. Viene inizializzato immediatamente dal costruttore a partire da appositi dati (ovvero specifiche credenziali AWS) riguardanti le proprietà dello User Pool del back-end in cui sono contenuti i dati di interesse per la gestione degli utenti.

Metodi:

1. + `getCognitoUsers()` :
`Promise<AWS.CognitoIdentityServiceProvider.ListUsersResponse>`
Metodo che restituisce al front-end l'elenco completo di tutti gli utenti dello User Pool di Amazon Cognito sottoforma di un oggetto di tipo `ListUsersResponse`.
2. + `createCognitoUsers(username: string, password: string, emailValue: string, phone_numberValue: string, sms: string, email: string)` : `string`
Metodo che consente di creare un nuovo utente a partire dai dati di tipo stringa passati come parametri. Gestisce anche i casi di errore restituendo una stringa che informa se la creazione del nuovo utente ha avuto successo oppure contenente la tipologia di errore verificatosi.
3. + `listGroup()` :
`Promise<AWS.CognitoIdentityServiceProvider.ListGroupsResponse>`
Metodo che restituisce al front-end l'elenco completo di tutti i gruppi dello User Pool di Amazon Cognito sottoforma di un oggetto di tipo `ListGroupsResponse`.
4. + `listGroupForUser(username : string)` :
`Promise<AWS.CognitoIdentityServiceProvider.AdminListGroupsForUserResponse>`

⁶Tutti i metodi di questa classe sono asincroni perché fanno a loro volta uso di metodi asincroni (che sono i metodi dell'SDK). Essi restituiscono oggetti tramite il meccanismo delle *promise* di Angular, che consente all'applicazione di continuare ad andare in esecuzione (senza dunque attendere l'arrivo del dato) anche se l'effettivo dato non è ancora stato restituito dal back-end remoto. Questo approccio migliora di molto la fluidità dell'applicazione con effetti positivi anche sulla *user experience*.

Metodo che restituisce al front-end l'elenco dei gruppi dello User Pool di Amazon Cognito relativi a un singolo utente (il cui nome viene passato come parametro a questo metodo) sottoforma di un oggetto di tipo `AdminListGroupsForUserResponse`.

5. + `getUserProperties(username : string) : Promise<AWS.CognitoIdentityServiceProvider.Admin GetUserResponse>`
Metodo che restituisce al front-end l'elenco di tutte le proprietà di un singolo utente (il cui nome viene passato come parametro a questo metodo) dello User Pool di Amazon Cognito.
6. + `addUserToGroup(groupName : string, username : string) : void`
Metodo che aggiunge il singolo utente identificato dal parametro `username` al gruppo dello User Pool di Amazon Cognito identificato dal parametro `groupName`.
7. + `removeUserFromGroup(groupName : string, username : string) : void`
Metodo che rimuove il singolo utente identificato dal parametro `username` dal gruppo dello User Pool di Amazon Cognito identificato dal parametro `groupName`.

UserPropertyRow

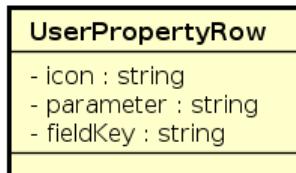


Figura 4.46: UserPropertyRow

Descrizione: Classe che si riferisce a una riga statica della lista di proprietà degli utenti.

Campi dati:

1. - `icon : string`
Stringa contenente l'identificativo dell'icona (del tema Inspinia).
2. - `parameter : string`
Stringa rappresentante un effettivo campo dati di un generico oggetto `User` a cui si riferisce la riga della lista di proprietà (tale stringa sarà identica al nome di un campo dati di un oggetto `User`).
3. - `fieldKey : string`
Stringa contenente la chiave del dato oggetto della riga (ovvero la sua descrizione statica).

UserDetailComponent

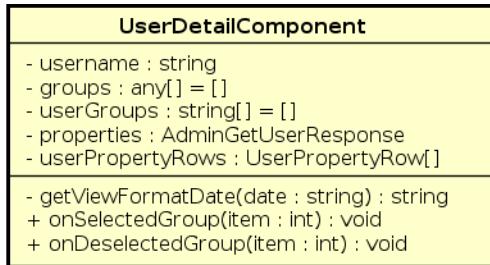


Figura 4.47: UserDetailComponent

Descrizione: Classe *component* di Angular per la rappresentazione della pagina web *Pagina di dettaglio di un singolo utente*.

Campi dati:

1. **- username : string**
Stringa che rappresenta il nome dello specifico utente associato alla pagina. Questo parametro viene recuperato dal costruttore grazie al meccanismo di routing con passaggio di parametri.
2. **- groups : any[]**
Array di oggetti anonimi (il formato di questi oggetti sarà una coppia *chiave-valore*) che rappresenta l'elenco di tutti i gruppi all'interno dello User Pool di Amazon Cognito.
3. **- userGroups: string[]**
Array di stringhe che si riferisce ai gruppi all'interno dello User Pool di Amazon Cognito relativi allo specifico utente associato alla pagina.
4. **- properties: Admin GetUserResponse**
Oggetto di tipo `Admin GetUserResponse` che rappresenta tutte le proprietà dello specifico associato alla pagina.
5. **- userPropertyRows: UserPropertyRow[]**
Array di oggetti di tipo `UserPropertyRow` che rappresenta l'insieme delle righe (ciascuna riga nel template è associata ad una proprietà dell'utente).

Metodi:

1. **- getViewFormatDate(date : string) : string**
Metodo che restituisce una stringa che rappresenta un orario (campo `date` passato come parametro) nel formato “GG:MM:YYYY - H:M:S”.
2. **+ onSelectedGroup(item: any)**
Metodo che viene invocato al click dell'utente sul nome di un gruppo all'interno del menu a tendina (casella di input a selezione multipla) e ha l'effetto di associare il gruppo selezionato allo specifico utente a cui è associata la pagina.
3. **+ onDeselectedGroup(item: any)**
Metodo che viene invocato quando un utente esegue un click sull'icona di “X” associata al nome di un gruppo (casella di input a selezione multipla) e ha l'effetto di dissociare il gruppo selezionato allo specifico utente a cui è associata la pagina.

CreateUserComponent

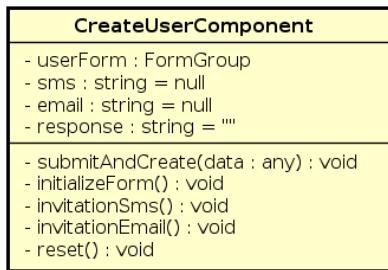


Figura 4.48: CreateUserComponent

Descrizione: Classe *component* di Angular per la rappresentazione della pagina web *Pagina di creazione di un nuovo utente*.

Campi dati:

1. - `userForm : FormGroup`
Oggetto di tipo `FormGroup` che si riferisce all'intera form (contenente tutti i vari campi dati di input) della pagina.
2. - `sms : string`
Stringa che si riferisce alla selezione della checkbox dell'SMS.
3. - `email : string`
Stringa che si riferisce alla selezione della checkbox dell'Email.
4. - `response : string`
Stringa che apparirà sul front-end al click del pulsante per la creazione di un nuovo utente. In caso di creazione effettiva del nuovo utente tale stringa conterrà il messaggio "User successfully created"; in caso contrario conterrà un messaggio che informerà l'utente sull'effettivo errore che ha impedito la creazione dell'utente.

Metodi:

1. - `submitAndCreate(data: any) : void`
Metodo invocato al click sul pulsante per la creazione di un nuovo utente. Questo metodo crea un nuovo utente grazie all'uso dello `UserService` a partire dal dato passato come parametro. Dopo la creazione (o il fallimento a causa di un errore) viene invocato automaticamente il metodo `reset()`.
2. - `initializeForm() : void`
Metodo invocato automaticamente dal costruttore che inizializza in una sola volta tutti i campi di input (grazie al *service FormBuilder*). Conterranno tutti la stringa vuota inizialmente e la casella di input per l'username sarà l'unica ad essere obbligatoria (sarà dichiarata `required` grazie all'uso della classe `Validator` delle *reactive forms* di Angular).
3. - `invitationSms() : void`
Metodo invocato automaticamente. Cambia il valore del campo dati `sms` a seconda che la checkbox corrispondente abbia o meno la spunta.

4. - invitationEmail() : void

Metodo invocato automaticamente. Cambia il valore del campo dati `email` a seconda che la checkbox corrispondente abbia o meno la spunta.

5. - reset() : void

Metodo che resetta tutti i campi di input della form facendoli tornare ai loro valori di default (con la visualizzazione delle stringhe placeholder) solo ed esclusivamente nel caso in cui non ci sia stato un errore in fase di creazione di un utente.

Capitolo 5

Test e Validazione

In questo capitolo verranno descritte le modalità dei test e i test stessi svolti al termine dell'attività di implementazione della soluzione progettata, che hanno permesso di effettuare la fase di verifica e di validazione del prodotto finale.

5.1 Modalità di esecuzione dei test

I test effettuati per garantire la qualità del prodotto rientrano tutti nella tipologia dei Test Funzionali. Un Test Funzionale è un test di tipo *black-box* ed è basato sulle specifiche funzionali delle componenti software. Preso un requisito funzionale, verifica come il sistema software implementa ed esegue le funzionalità a cui esso si riferisce. Ogni insieme di test è stato progettato al termine di ogni piccolo ciclo di sviluppo (ovvero dopo la fase di codifica per ogni pagina web assegnata) come concordato con il Tutor Aziendale.

5.2 Test

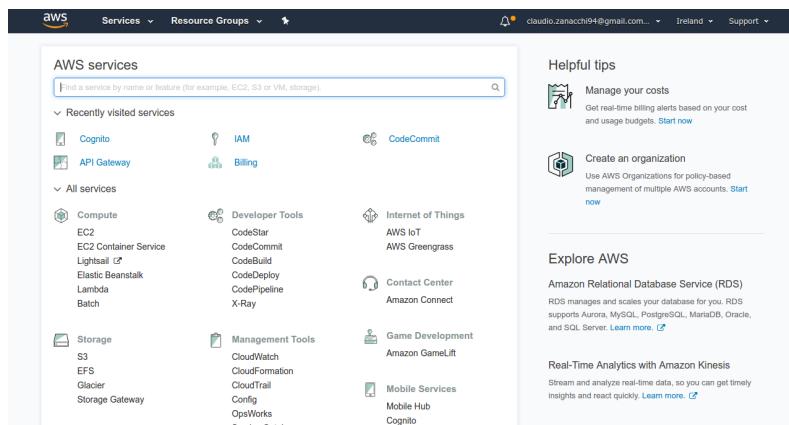


Figura 5.1: Console AWS

Tutte le modifiche ai dati presenti nel back-end dell'applicazione durante la fase di testing sono state effettivamente riscontrate osservando direttamente tali dati

modificati per mezzo della **Console AWS** (ovvero la console di gestione che consente di utilizzare i vari servizi messi a disposizione da AWS direttamente da un browser web e monitorare lo stato dei dati gestiti da questi).

Direttamente dal browser è stato infatti possibile riscontrare che dopo le interazioni con il front-end effettuate tramite le componenti dell'applicazione realizzate, ci sono stati i cambiamenti attesi dei dati sia all'interno del database DynamoDB, sia all'interno dello User Pool di Amazon Cognito.

Particolare rilevanza hanno anche avuto gli “strumenti per sviluppatori” del browser Google Chrome per i test sul *design responsive* delle pagine web assegnate. Un requisito comune a tutte le pagine web assegnate era che esse dovessero adattarsi graficamente in modo automatico rispetto al dispositivo con le quali venivano visualizzate, riducendo dunque la necessità dell'utente di ridimensionare e scorrere i contenuti, migliorando di gran lunga la sua esperienza.

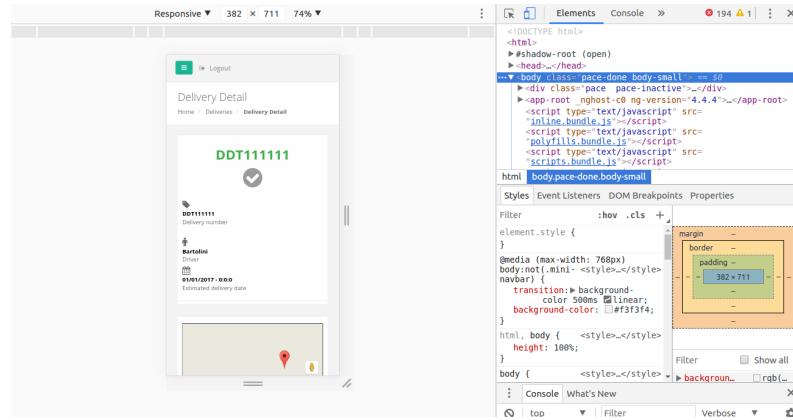


Figura 5.2: Esempio 1 di test per il design responsive

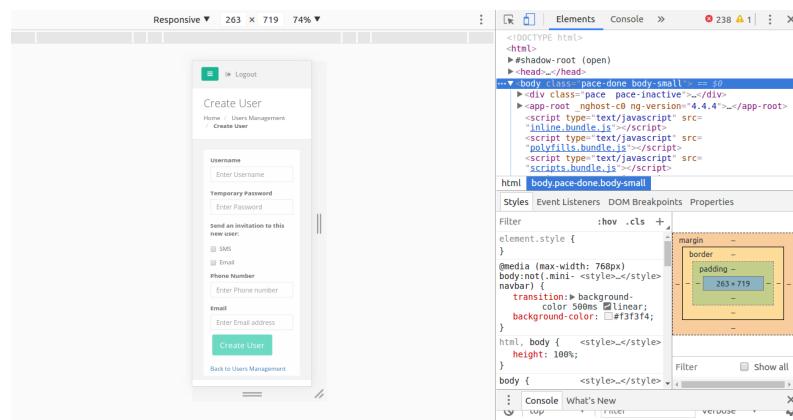


Figura 5.3: Esempio 2 di test per il design responsive

Seguirà una tabella contenente codice, descrizione ed esito dei test più rilevanti eseguiti.

Ogni test sarà identificato univocamente da un **Codice**. Esso è espresso nella seguente notazione:

[Ambito del test]-[Numero univoco]

Dove per **Ambito del test** si intende:

- * **DC**: Pagina di dettaglio di una consegna
- * **LU**: Pagina della lista degli utenti
- * **NU**: Pagina di creazione di un nuovo utente
- * **DU**: Pagina di dettaglio di un singolo utente

Dalla Tabella 5.1 della pagina seguente si può notare che tutti i test hanno avuto degli esiti positivi. I risultati attesi di ogni test si sono dunque verificati ed essi hanno coperto e verificato tutti i requisiti funzionali (il cui elenco completo è visibile nella tabella 3.1).

Tabella 5.1: Tabella dei Test

Codice	Descrizione	Esito
DC-1	All'interno della tabella nella <i>Pagina della lista delle consegne</i> , un click su una qualsiasi riga deve far aprire la <i>Pagina di dettaglio di una consegna</i> relativa esattamente alla consegna indicata nella riga su cui è stato fatto click	Positivo
DC-2	Al click sulla foto della bolla di consegna si apre una <i>lightbox</i> che mostra la foto nelle sue dimensioni originali	Positivo
DC-3	Deve essere possibile utilizzare l'applicazione di Google Maps usufruendo di tutte le sue funzionalità direttamente dentro la pagina web	Positivo
DC-4	Un click sull' icona a forma di matita (che appare spostando il cursore del mouse accanto all'icona esplicativa di una proprietà) deve far comparire una casella di input al posto del valore della proprietà	Positivo
DC-5	Un click sul pulsante a forma di "X" dopo l'apertura della casella di input deve fare sparire la casella di input e la singola proprietà proprietà deve tornare allo stato precedente del click sull'icona a forma di matita	Positivo
DC-6	Un click sul pulsante a forma di spunta dopo l'apertura della casella di input non deve produrre alcun risultato se non è stato inserito nessun dato nella casella di input	Positivo
LU-1	Un click sul pulsante "Create new user" deve far aprire la <i>Pagina di creazione di un nuovo utente</i>	Positivo
LU-2	Un click su una riga della tabella deve far aprire la <i>Pagina di dettaglio di un singolo utente</i> relativa esattamente all'utente indicato nella riga su cui è stato fatto click	Positivo
LU-3	Il passaggio del cursore del mouse su una riga della tabella deve metterla in evidenza rispetto alle altre righe cambiandone il colore	Positivo
LU-4	Digitare lettere all'interno di una qualsiasi delle caselle di input dovrà ridurre dinamicamente il numero di righe della tabella	Positivo
NU-1	Un click sul pulsante "Create user" non deve produrre alcun risultato se non è stato inserito nessun dato nella casella di input relativa all'username	Positivo
NU-2	Deve sempre essere possibile tornare alla <i>Pagina della lista degli utenti</i> tramite un click sul link "Back to Users Management"	Positivo
NU-3	Deve sempre apparire la stringa "User successfully created" al click del pulsante "Create user" se sono stati inseriti nelle caselle di input dati validi. Inoltre tutte le caselle di input dovranno essere resettate e tornare allo stato iniziale	Positivo
NU-4	Deve sempre apparire una stringa contenente un messaggio di errore se nelle caselle di input sono stati inseriti dati non validi. La stringa cambia in base all'errore commesso	Positivo
DU-1	Deve sempre aprirsi un menu a tendina contenente i nomi di tutti i gruppi dello User Pool associati all'applicazione al click sulla casella di input a selezione multipla	Positivo
DU-2	Un click su un qualsiasi gruppo nel menu a tendina deve far chiudere il menu a tendina e aggiungere questo gruppo all'interno della casella di input a selezione multipla	Positivo
DU-3	Un click sulla piccola icona a forma di "X" posta accanto al nome di un gruppo all'interno della casella di input a selezione multipla deve rimuovere questo gruppo dalla casella	Positivo
DU-4	Deve sempre essere possibile tornare alla <i>Pagina della lista degli utenti</i> tramite un click sul link "Back to Users Management"	Positivo
DU-5	In base all'utente a cui si riferisce la pagina, il numero di proprietà identificate dall'icona di una matita deve essere variabile	Positivo

Capitolo 6

Valutazione retrospettiva

In questo capitolo finale viene fatto il punto della situazione riguardo il conseguimento degli obiettivi prefissati alla fine dell'attività di stage, un consuntivo orario finale che si riallaccia al piano di lavoro inizialmente concordato, una descrizione delle problematiche incontrate e delle nuove conoscenze acquisite e infine una valutazione sull'intero corso di laurea.

6.1 Raggiungimento degli obiettivi prefissati

Durante l'attività di stage qualche requisito riguardante alcune delle funzionalità pianificate è stato cambiato in corso d'opera, portando a un allungamento delle fasi di progettazione di implementazione.

Per mancanza di tempo non è stato purtroppo possibile la lavorazione inizialmente pianificata riguardante la progettazione e implementazione di *Dashboard di monitoraggio con indicatori riassuntivi*.

La fase 5 del piano di stage, ovvero *Sviluppo e integrazione di procedure di test End-To-End per le funzionalità dell'applicazione* è stata eliminata dal piano sempre per motivi legati alle tempistiche, ma tale fase era stata etichettata fin dall'inizio come opzionale e difficilmente fattibile in tempo anche a detta del Tutor Aziendale.

Inoltre la strategia adottata di utilizzare piccoli cicli di sviluppo (uno per ogni componente dell'applicazione assegnata) ha sicuramente semplificato la metodologia, ma ha anche fatto sì che non si seguisse in modo del tutto lineare la netta divisione in fasi del piano di lavoro concordato inizialmente.

In generale il lavoro svolto è stato apprezzato dal Tutor Aziendale e mi ritengo comunque appagato dal risultato finale, poiché i requisiti più importanti (ovvero quelli individuati nel Paragrafo 3.5) sono stati quasi tutti soddisfatti. Di seguito è riportata una tabella riassuntiva (in tabella non sono indicate distinzioni tra requisiti funzionali e non funzionali):

Requisiti	Individuati	Soddisfatti
Totali	49	46
Obbligatori	23	23
Desiderabili	21	19
Opzionali	5	4

Come si può notare il 100% dei requisiti obbligatori è stato soddisfatto.
Per quanto riguarda i requisiti desiderabili, quelli a non essere stati soddisfatti sono stati:

- * **RFD-6** (solo un utente amministratore può modificare alcuni dati di una consegna), in quanto non è mai stato realmente progettato (al momento) un sistema di autenticazione a diversi livelli di privilegi per chi esegue il login nell'applicazione. I campi dati di una consegna sono quindi modificabili da qualsiasi utente che abbia effettuato l'autenticazione.
- * **RFD-25** (La navigazione deve essere implementata tramite una breadcrumb dinamica e non statica) perchè è stato ritenuto troppo complesso utilizzare un'unica breadcrumb (tramite una classe *component* di Angular) che si adattasse dinamicamente ad ogni pagina dell'applicazione. È stato ritenuto opportuno dunque utilizzare una breadcrumb statica (il cui codice HTML veniva scritto direttamente nel template del relativo *component* riferito alla pagina web) perchè implementabile molto più facilmente e perchè era già una funzionalità inclusa nel tema grafico Inspinia.

I requisiti opzionali a non essere stati soddisfatti sono stati:

- * **RFF-10** (Le fotografie del dettaglio della riserva possono essere più di una) perchè al momento i dati del back-end dell'applicazione erano stati progettati per avere soltanto un unico riferimento alla foto del dettaglio di una riserva. Ciononostante questa unica foto (nel caso di consegna con riserva) viene visualizzata come unica foto in una galleria di immagini. Questa scelta è stata fatta per favorire l'integrazione all'interno di questa galleria di altre immagini in futuro.

6.2 Consuntivo orario finale

Rispetto al piano di lavoro di inizio stage concordato con il Tutor Aziendale, ci sono state alcune variazioni relative ai giorni di lavoro (un diagramma di Gantt dei giorni lavorativi concordati inizialmente è presente nel Capitolo 1). Tali variazioni non hanno influito però sulla durata dello stage in termini di ore lavorative, che sono rimaste comunque 312.

I giorni del 19, 20 e 21 luglio erano stati esclusi dal piano di stage fin dall'inizio (sono stati infatti considerati come giorni aggiuntivi per facilitare la parte di apprendimento delle tecnologie).

Era prevista la fine del periodo di stage per giorno **22 settembre** (in quanto c'è stato il periodo di ferie nei giorni della settimana dal 14 al 18 agosto, che non sono stati ovviamente contati).

A causa di diversi impegni, sia personali che legati all'università, sono risultato assente nei seguenti cinque giorni:

- * Lunedì 21 agosto

- * Giovedì 31 agosto
- * Mercoledì 13 settembre
- * Giovedì 14 settembre
- * Martedì 19 settembre

Ciò ha fatto sì che il mio stage si concludesse giorno **29 settembre** (ovvero con una settimana lavorativa di calendario in più rispetto al tempo stabilito inizialmente). Questo non ha costituito un problema in quanto la scadenza ultima della copertura assicurativa era stata concordata proprio per giorno 29 settembre.

6.3 Problematiche incontrate

Non avendo mai lavorato ad una applicazione web fino ad ora (ma soltanto su pagine web statiche) inizialmente imparare ad utilizzare il framework Angular si è dimostrato abbastanza impegnativo e ha richiesto più tempo del previsto. Per fortuna tale framework gode di un'ottima e chiara documentazione (oltre che di un intuitivo tutorial) che si è alla fine dimostrata più che adatta agli scopi che erano stati prefissati.

Ho trovato che la scelta fatta dall'azienda di utilizzare un tema già esistente per la creazione dello stile grafico delle pagine web piuttosto che di utilizzare uno stile grafico totalmente libero in mano allo sviluppatore abbia qualche volta creato alcuni problemi durante la fase di implementazione (principalmente legati a conflitto tra diverse regole di presentazione incompatibili).

Un altro problema che ho riscontrato è stata un po' di carenza nel lavoro di gruppo con gli altri sviluppatori al lavoro sulle altre parti della stessa applicazione. Sono convinto che se ci fossero state più occasioni di dialogo (per esempio più riunioni specifiche su questo progetto) avrei compreso molto più rapidamente alcuni requisiti che erano indissolubilmente legati al lavoro svolto in parallelo dagli altri sviluppatori. Mi sono trovato quindi qualche volta a modificare ciò che stavo facendo in corso d'opera e questo si sarebbe potuto evitare con una migliore coordinazione tra i vari membri del team (e ciò avrebbe comportato anche un risparmio in termini di tempo).

6.4 Conoscenze acquisite

Mi ritengo molto soddisfatto di tutte le conoscenze (sia teoriche che pratiche) acquisite nei soli due mesi di stage.

Per cominciare ho ritenuto di grande valenza l'aver esplorato da solo il funzionamento a tutto tondo di una applicazione web, studiandone quasi ogni singola componente (sia del front-end che del back-end). Questo mi ha aiutato a capire bene come avviene realmente una interazione tra questi livelli e anche concetti molto importanti come la *Separation of concerns*, che fino ad ora avevo semplicemente studiato in teoria.

Ho ritenuto molto interessante e anche divertente lavorare con Angular 4 e sono sicuro che lo userò ancora in futuro per creare applicazioni web performanti. Il suo studio ha anche rafforzato le mie conoscenze su linguaggi come CSS3 (che non utilizzavo da molto tempo) e mi ha permesso di imparare un linguaggio nuovo.

La sintassi del linguaggio TypeScript si è dimostrata molto semplice e intuitiva e mi ha permesso di produrre codice di qualità, scritto rispettando i paradigmi della programmazione orientata agli oggetti (cosa che sarebbe risultata di gran lunga più difficile utilizzando JavaScript).

Sono anche molto contento di aver appreso il funzionamento dei più famosi servizi di AWS, sia perché per ciò che un informatico deve fare essi sono indubbiamente utili, sia perché sono tecnologie all'avanguardia che stanno prendendo piede molto rapidamente (sono oggi il riferimento sul mercato del *cloud computing* numero uno¹).

6.5 Valutazione personale sullo stage

Questa esperienza di stage nel suo complesso è stata positiva. Ho ritenuto molto accogliente e sereno l'ambiente di lavoro (totalmente open space) e molto disponibile il Tutor Aziendale.

Ho scelto un ambito legato allo sviluppo di una applicazione web in quanto lo ritengo un aspetto molto interessante e attuale, e anche per il fatto che questo ambito lega insieme veramente tanti concetti spiegati in corsi universitari apparentemente diversi tra loro (dai paradigmi di pura programmazione, al funzionamento delle tecnologie del web, dai protocolli di rete internet al puro design grafico ecc...).

Come già detto nei precedenti paragrafi ho ritenuto molto interessanti e utili le tecnologie proposte e sono molto contento di ciò che ho appreso. Trovo che questa esperienza “sul campo” sia stata molto più formativa e preziosa per il mio futuro rispetto alla frequenza di un qualsiasi corso universitario.

Ritengo però che la durata di soli due mesi per lo stage obbligatorio sia un arco di tempo davvero troppo breve, sia per formare uno stagista inesperto, sia per portare a compimento un progetto minimamente impegnativo: ho avuto infatti la sensazione di aver appreso il funzionamento essenziale delle tecnologie proposte soltanto al termine dello stage stesso e mi sarebbe piaciuto avere più tempo per completare tutti gli obiettivi proposti nell'originale piano di lavoro (mi sono reso conto che il lavoro sulle ultime componenti assegnate da parte mia è stato di gran lunga più rapido rispetto a quelle assegnate per prime).

Sono convinto anche di aver studiato (durante la fase iniziale di formazione) forse in modo fin troppo approfondito le funzionalità dei servizi di AWS legati al back-end dell'applicazione rispetto poi all'uso che ne ho dovuto fare concretamente. Per esempio mi sarebbe piaciuto lavorare anche sul codice delle funzioni Lambda utilizzando il linguaggio di programmazione Java (che è un linguaggio che ritengo di conoscere abbastanza bene). Questo mi avrebbe aiutato a capire ancora meglio il funzionamento globale del back-end dell'applicazione.

¹Per una descrizione più accurata di questo fenomeno si rimanda alla Appendice C

6.6 Valutazione personale sul corso di studi

Iscritto al Corso di Laurea in Informatica per pura curiosità (non avendo avuto in precedenza particolari passioni viscerali per il settore), a diversi anni da questa scelta quasi casuale mi ritengo pienamente soddisfatto della strada da me intrapresa. Ho avuto effettivamente modo di confrontarmi con molti altri studenti provenienti da diversi corsi di studi e sono assolutamente certo di aver frequentato uno dei corsi in assoluto più esaustivi e soprattutto utili alla preparazione professionale nel mondo del lavoro.

Trovo che la scelta di inserire all'interno del corso una grossa quantità di impegnativi progetti didattici (oltre che lo stage finale obbligatorio) lo renda effettivamente molto diverso dalla maggior parte dei corsi universitari di oggi. Ho realizzato che tali progetti sono stati effettivamente il modo migliore e più efficace per imparare a capire davvero che cosa si stesse studiando e come metterlo in pratica.

Ho trovato però i contenuti del corso riguardanti le basi matematiche non molto curati e poco interessanti. Essi sono effettivamente indispensabili, ma per come sono impostati li reputo fine a se stessi e soprattutto non ben spalmati durante i tre anni. Ritengo che una maggiore integrazione progressiva nel tempo tra materie di matematica e materie di informatica sia un vantaggio perché aiuta la memoria degli studenti ad essere sempre allenata su dei concetti che altrimenti andrebbero messi da parte.

Particolarmente rilevanti per l'attività di stage sono stati i corsi di *Tecnologie web* (che mi ha insegnato ad avere padronanza con le principali tecnologie di riferimento come HTML, CSS e JavaScript, oltre che una spiccata conoscenza dell'accessibilità dei siti web) e *Web information management* (per tutti i concetti legati alla *web usability*).

Infine, mi sento in dovere di fare una menzione speciale ai tre corsi *Reti e sicurezza*, *Programmazione ad oggetti* ed *Ingegneria del software*, che ho reputato in assoluto i più ben fatti e utili dell'intero corso di studi. Ovviamente anche le nozioni spiegate in questi corsi sono state particolarmente rilevanti durante l'intero periodo di stage.

Appendice A

Verbali

Questa appendice include tutti i verbali delle riunioni avvenute in azienda con il Tutor Aziendale. Ogni riunione ha avuto forte influenza sullo sviluppo del progetto di stage in corso d'opera.

Verbale 19-07-2017

Descrizione della vecchia applicazione *Logistic Performance*

Il Tutor Aziendale ha illustrato in modo esauriente l'architettura e le funzionalità dell'applicazione *Logistic Performance* creata da IKS. Essa verrà presa come punto di riferimento durante l'intero stage, proprio perché essendo il progetto una totale reimplementazione della stessa applicazione su nuove tecnologie, saranno molti i requisiti ottenuti a partire dalle vecchie funzionalità di questa prima versione dell'applicazione.

Descrizione del progetto di stage

Il progetto di stage si è rivelato essere un progetto in ottica di sperimentazione tecnologica. È stato inoltre ufficializzato il fatto che dopo la prima fase di autoapprendimento delle tecnologie, la prima componente dell'applicazione su cui lavorare sarebbe stata la *Pagina di dettaglio di una consegna*. Il software scelto per il coordinamento in azienda è stato Slack.

Definizione dei primi obiettivi

Il tutor aziendale ha sottolineato l'importanza della prima fase di autoapprendimento delle tecnologie. Ha inoltre offerto il suo aiuto ognqualvolta si dovessero incontrare delle difficoltà. Sono stati consigliati i seguenti tutorial per le tecnologie:

- * *Tour of heroes* per Angular 4 (<https://angular.io/tutorial>)
- * *Getting Started with AWS* per AWS (<https://aws.amazon.com/it/documentation/gettingstarted>)

Verbale 27-07-2017

Descrizione dell'applicazione allo stato attuale

Il Tutor Aziendale ha fatto il punto sulla situazione in cui si trova l'applicazione *Logistic Performance* allo stato attuale. Essenzialmente ci sono tre *repository* in cui si trovano i file di codice sorgente della parte di front-end, della parte di back-end e della versione mobile dell'applicazione (quest'ultimo contiene anche dei mockup creati da un altro stagista). I servizi di back-end sono esposti verso l'esterno tramite *API Gateway* e queste API sono generate a partire da un file *Swagger* (che è una specifica che documenta le API HTTP/REST). Altri programmatore sono al lavoro su pipeline automatiche che eseguono automaticamente compilazione e deploy ad ogni commit effettuato sul repository.

Strategia del ciclo di sviluppo adottata per la componente assegnata

Concordato ufficialmente l'approccio al ciclo di sviluppo per la realizzazione della *Pagina di dettaglio di una consegna*, ma che potrebbe risultare efficace anche per le prossime parti dell'applicazione che verranno assegnate in futuro. Esso sarà diviso nelle seguenti fasi:

- * Analisi dei requisiti: fase in cui vengono definite le funzionalità della pagina e poi raccolti i requisiti;
- * Realizzazione di wireframe e mockup;
- * Studio delle componenti di back-end (servizi AWS): fase necessaria per il soddisfacimento dei requisiti;
- * Eventuale modifica alle componenti di back-end: potrà essere svolta direttamente su codice sorgente Java delle funzioni Lambda o con gli strumenti messi a disposizione della console AWS. Questa è una fase opzionale in quanto essendo un progetto di gruppo, potrebbero essere già state implementate delle funzioni di back-end adatte alle specifiche circostanze;
- * Progettazione di dettaglio del front-end;
- * Implementazione;
- * Verifica tramite test;

Introduzione agli strumenti di supporto

Per la creazione dei wireframe e dei mockup è stato concordato l'uso del di *Adobe Experience Design*, software di Adobe per creare grafiche per la *User Experience* ancora in versione Beta, e pertanto scaricabile gratuitamente. Il Tutor Aziendale ha suggerito di prendere spunto dai mockup già realizzati della versione mobile in modo che l'applicazione possa avere uno stile grafico coerente su diversi dispositivi. Lo stile grafico dell'applicazione non dovrà essere creato interamente, ma verrà sempre utilizzato un tema grafico già esistente, ovvero *Inspinia*, che è un tema *responsive*, graficamente accattivante e in linea con i tempi. In questo tema sono presenti moltissimi stili e componenti (come tabelle, form e icone) che si possono comodamente riutilizzare. Tutto il codice sorgente prodotto in fase di implementazione dovrà essere di buona qualità e dovrà essere versionato sul repository Git tramite il servizio *AWS CodeCommit*.

Definizione della documentazione in output

Concordata ufficialmente la documentazione in uscita: essa sarà costituita dai documenti *Analisi dei requisiti* e *Specifica tecnica* (quest'ultima sarà redatta subito prima dell'implementazione).

Verbale 08-08-2017

Mockup a diversi livelli di astrazione

Con il Tutor Aziendale è stato concordato l'approccio alla progettazione dei mockup a diversi livelli di astrazione progressivi, dal meno dettagliato al più dettagliato (precisamente Wireframe, Wireframe con dettaglio delle aree, Mockup, Mockup della User Interface).

Requisiti della *Pagina di dettaglio di una consegna*

Il Tutor Aziendale ha approvato la maggior parte dei requisiti scaturiti dalla prima fase di analisi. Ciononostante sono state concordate diverse variazioni di alcuni requisiti e ne sono stati fissati degli altri.

Progettazione architetturale della *Pagina di dettaglio di una consegna*

Per quanto riguarda la fase di progettazione architetturale delle componenti software, il Tutor Aziendale ha caldamente consigliato un approccio volto alla rimozione delle dipendenze, e dunque del disaccoppiamento delle componenti ove possibile. Questo approccio facilita di molto la manutenibilità del prodotto.

Verbale 22-08-2017

Approvazione dei mockup

Il Tutor Aziendale ha approvato i mockup realizzati in fase di prototipizzazione dell'interfaccia utente dichiarandosi soddisfatto del risultato.

Progettazione di dettaglio della *Pagina di dettaglio di una consegna*

Il Tutor Aziendale ha approvato la maggior parte della logica di business progettata per il soddisfacimento dei requisiti previsti. Ha inoltre consigliato fortemente di garantire il *Single Responsibility Principle* sia in fase di progettazione delle classi che in fase di progettazione dei metodi.

Istruzioni per il versionamento del codice

È stato indicato esplicitamente il repository sul quale effettuare i commit di codice sorgente. È stato inoltre confermato che il software usato per il versionamento del codice dovrà essere *AWS CodeCommit*, software di AWS per la gestione di repository Git privati. Il Tutor Aziendale ha fortemente consigliato un approccio *self-contained*, ritenendo opportuno e più efficiente effettuare molti piccoli commit e farlo di frequente, piuttosto che effettuare pochi commit di grandi dimensioni in un arco di tempo più ampio.

Modalità dei test funzionali

La scelta di utilizzare il browser *Google Chrome* per testare il funzionamento dell'applicazione è stato approvato dal Tutor Aziendale. L'uso dei *Developer Tools* di Chrome infatti, facilita il ritrovamento di errori nel codice sorgente.

Verbale 04-09-2017

Nuovi requisiti assegnati per la *Pagina di dettaglio di una consegna*

Sono stati assegnati direttamente dal Tutor Aziendale dei nuovi requisiti per la *Pagina di dettaglio di una consegna*; essi riguardano:

- * L'implementazione di un meccanismo di *editing* della lista delle proprietà della specifica consegna oggetto della pagina (non essendoci un sistema di gestione degli utenti al momento tale modifica dei campi potrà essere effettuata da qualsiasi utente dell'applicazione).
- * Modificare la *Pagina della lista delle consegne*, ovvero la pagina che mostra (tramite una tabella) l'elenco di tutte le consegne effettuate e non presenti nell'applicazione. Nonostante la modifica di questa pagina non fosse contemplata negli obiettivi dello stage, è stato ritenuto opportuno introdurre il requisito della navigazione (con un click su una riga di questa tabella da parte dell'utente) da questa pagina alla *Pagina di dettaglio di una consegna*, in quanto è stato stabilito che questo nuovo il requisito riguarda comunque l'ambito della *Pagina di dettaglio di una consegna*.
- * Cercare il più possibile di scrivere codice volto ad automatizzare la visualizzazione della lista di proprietà di una consegna: è stato suggerito di seguire un approccio basato sulla creazione di classi logiche il cui scopo sarebbe stato proprio quello di rappresentare un certo tipo di oggetto grafico della pagina (in questo caso una classe volta a rappresentare graficamente una qualsiasi riga della lista di proprietà avrebbe aiutato). Questo approccio semplifica notevolmente il template HTML della pagina web, che in questo modo non è più eccessivamente lungo a causa di istruzioni simili ripetute (in questo caso una istruzione per ogni riga della lista).

Verbale 07-09-2017

Punto della situazione sullo stato dell'arte dell'applicazione

In vista di una prossima demo sulla parte di front-end dell'applicazione da fare vedere al cliente, è stato fatto il punto della situazione su vari aspetti dell'applicazione *Logistic Performance* con tutte le persone che momentaneamente erano a lavoro proprio su questi aspetti, tra cui:

- * L'implementazione delle funzionalità di autenticazione;
- * La sistemazione della *Pagina della lista delle consegne*. In particolare è stato ritenuto importante indicare la tipologia della consegna (attraverso l'apposito colore scelto per ogni tipologia) anche in questa pagina, essendo ormai stato definito il metodo `getDeliveryStatus()` della classe `Delivery`.

Verbale 18-09-2017

Descrizione ad alto livello del servizio Amazon Cognito

Il Tutor Aziendale ha descritto le principali funzionalità del servizio Amazon Cognito di AWS. Esso servirà per la gestione dei dati relativi agli utenti dell'applicazione e sarà fondamentale interagire con questo servizio direttamente dal front-end. È stato sottolineato che ciò che fa Cognito è fondamentalmente evitare allo sviluppatore di inventare da zero una tipologia di sistema per la gestione degli utenti (come fanno la maggior parte delle applicazioni) fornendogli direttamente un meccanismo di gestione completa.

Assegnamento di tre nuove pagine web

Sono state assegnate come nuovi obiettivi dello stage le tre pagine *Pagina della lista degli utenti*, *Pagina di creazione di un nuovo utente* e *Pagina di dettaglio di un singolo utente*. La lavorazione a ciascuna di queste tre pagine dovrà seguire il solito ciclo di sviluppo concordato inizialmente. Inoltre ciascuna pagina dovrà obbligatoriamente interfacciarsi al back-end per recuperare e modificare i dati presenti all'interno dello User Pool di Amazon Cognito.

SDK JavaScript al posto di Amazon API Gateway

Diversamente dalle altre pagine dell'applicazione, per l'interfaccia tra front-end e back-end di queste tre pagine web è stato concordato con il Tutor Aziendale un approccio alternativo: se prima veniva utilizzato il servizio Amazon API Gateway con metodi che nel front-end contenevano delle chiamate HTTP che si preoccupavano di interagire con le API esposte da questo servizio (Amazon API Gateway a sua volta invocava delle funzioni Lambda, che gli restituivano i dati provenienti da DynamoDB ed S3), adesso verrà usato un SDK JavaScript contenente moltissimi metodi molto utili per il recupero e la modifica di tutti i dati contenuti nello User Pool di Amazon Cognito.

Discussione sui test funzionali

Il Tutor Aziendale ha sottolineato l'importanza della progettazione dei pacchetti di test funzionali per la verifica operativa al termine del periodo di implementazione di queste nuove pagine web assegnate.

Verbale 20-09-2017

Approvazione dei mockup finali

Il Tutor Aziendale si è dichiarato soddisfatto di tutti i mockup finali da me realizzati in fase di prototipizzazione dell'interfaccia utente, concordando sulle numerose scelte progettuali fatte.

Cambiamento di un requisito per la *Pagina della lista degli utenti*

È stato concordato un cambiamento riguardante il requisito della ricerca degli utenti per parola chiave: inizialmente si era pensato a un unico campo di input posto in cima alla pagina web in cui era possibile inserire l'apposita parola chiave da ricercare; in seguito è risultato molto più agevole l'utilizzo di più campi di input (uno per ogni

tipologia di parola chiave) da ricercare. Questa scelta è stata fatta perché da un lato la pagina risultava così molto più coerente con i contenuti della *Pagina della lista delle consegne*, dall'altro lato risultava molto più semplice da implementare.

Verbale 29-09-2017

Verifica e validazione delle componenti assegnate

È stato mostrato al Tutor Aziendale che tutte le componenti dell'applicazione sviluppate soddisfano effettivamente i requisiti assegnati originariamente tramite gli appositi test funzionali da me progettati in fase di verifica. In generale il Tutor Aziendale si è dimostrato soddisfatto delle funzionalità correttamente implementate.

Discussione riguardante lo stage nel suo complesso

Il Tutor Aziendale ha avviato una discussione riguardante un mio feedback su vari aspetti positivi e negativi riscontrati sullo stage ormai giunto al termine.

Appendice B

L’evoluzione del framework Angular



Figura B.1: Logo di Angular

JavaScript, linguaggio di programmazione nato come semplice linguaggio di scripting per la creazione di effetti dinamici all’interno di pagine web, può oggi vantare una ampissima gamma di framework opensource.

Angular è dunque nato come un framework basato su JavaScript per consentire lo sviluppo di applicazioni web performanti (le Single Page Application ne sono un esempio).

La prima versione, nota come **AngularJS**, è stata ideata da Miško Hevery nel 2009. Hevery, essendo in seguito assunto da Google, fece in modo che l’intero progetto venisse preso in carico dell’azienda (ed è innegabile che uno dei motivi legati al successo di Angular sia stato effettivamente il supporto di Google). AngularJS fu rilasciato nel 2012 ed ebbe fin da subito un buon successo dovuto principalmente all’infrastruttura fornita. Essa infatti praticamente spingeva lo sviluppatore ad organizzare il codice separandolo sia a livello funzionale (layer strutturale scritto in HTML, layer di presentazione scritto in CSS e layer comportamentale scritto in JavaScript), che a livello logico con la separazione delle funzionalità in diversi *component*.

Sono veramente molte le funzionalità di supporto che sono state migliorate e introdotte dal framework, solo per citarne alcune:

- * Un ottimo supporto ai pattern *Dependency Injection* ed *MVC*;
- * Il concetto di *separation of concerns* (lo sviluppatore viene aiutato dal framework stesso a separare il codice in base ad ogni diversa funzionalità);
- * Il *two-way data binding*, ovvero quel meccanismo che consente di effettuare un collegamento diretto a due via dall'interfaccia al modello di business dei dati e viceversa, garandendo sempre e comunque la visualizzazione effettiva da parte dell'utente dei dati del modello sottostante (e mai di un insieme di dati non aggiornato);
- * Il concetto dei *services*, ovvero specifiche classi *Singleton*_G volte a implementare delle funzionalità (che tipicamente recuperano dati da componenti remoti) condivise da diverse classi della applicazione;
- * Il concetto di *components*, ovvero parti dell'interfaccia web, ciascuna con una propria struttura, aspetto e comportamento autonomo, che possono comunicare tra loro ed anche essere innestate l'una dentro l'altra;
- * Il supporto alla modularità e riusabilità dei componenti;
- * La definizione di un nuovo meccanismo di *routing*, ovvero ciò che consente a un utente di navigare tra due viste (in forma di pagine web) utilizzando URL e collegamenti ipertestuali;
- * Il supporto al *testing* delle applicazioni.

Nell'anno 2014 venne annunciata ufficialmente la versione 2 di Angular, che sarebbe stata chiamata **Angular 2**. L'annuncio di questa nuova versione provocò scalpore tra le comunità di sviluppatori, in quanto i creatori del framework hanno optato per una riscrittura completa di tutto il software, senza quindi garantire alcun tipo di retrocompatibilità con la versione precedente (che aveva ormai preso largamente piede). Tale scelta è stata fatta, a detta degli sviluppatori, perché era il miglior modo possibile di fornire agli sviluppatori un set di strumenti all'avanguardia per la creazione di applicazioni web. Questa seconda versione, pur non avendo quindi alcuna retrocompatibilità con la precedente, avrebbe in ogni caso mantenuto inalterati i concetti chiave alla base del framework, cambiando soltanto il modo pratico di utilizzarli.

Angular 2 introdusse in ogni caso molte novità; per esempio è stato di gran lunga migliorato il supporto ad eventi *touch*, le prestazioni sono molto migliorate grazie ad un uso più intelligente di alcune funzionalità (per esempio il *two-way data binding* non è stato più impostato come predefinito, ma opzionale per l'utente, in quanto era abbastanza dispendioso in termini di prestazioni), ed è stato introdotto il supporto al linguaggio di programmazione TypeScript, super-set di JavaScript, che prevede un sistema di controllo per i tipi e la cui sintassi è molto più simile a quella dei classici linguaggi ad oggetti come C++ e Java. Questo ha permesso uno sviluppo più facile e consapevole della logica di business di *components* via via sempre più complessi.

A partire dalla versione numero 2 del framework, i successivi aggiornamenti si sono fatti via via più frequenti e hanno mantenuto la retrocompatibilità (i creatori hanno assicurato che le versioni successive considereranno soltanto i miglioramenti del framework ed implementazioni di nuove funzionalità).

Dalla versione 2 del 2014 si è passati direttamente alla versione 4 del 2016, nota come **Angular 4** (la versione 3 è stata saltata a causa di un disallineamento^[5]). Per finire,

```

export class Destination {

    public street: string;
    public country: string;
    public city: string;
    public state: string;
    public postalCode: string;

    constructor(street: string,
               country: string,
               city: string,
               state: string, postalCode:
               string) {
        this.street = street;
        this.country = country;
        this.city = city;
        this.state = state;
        this.postalCode =
            postalCode;
    }
}

```

```

define(["require", "exports"],
function (require, exports) {
    "use strict";
    Object.defineProperty(
        exports,
        "__esModule",
        { value: true });
    var Destination =
        (function () {
            function Destination(
                street,
                country,
                city, state,
                postalCode) {
                this.street = street
                    ;
                this.country =
                    country;
                this.city = city;
                this.state = state;
                this.postalCode =
                    postalCode;
            }
            return Destination;
        })();
    exports.Destination =
        Destination;
});

```

Figura B.2: Confronto tra sintassi TypeScript e sintassi JavaScript di codice equivalente

proprio nel 2017 è stata rilasciata la versione 5 dell'applicazione.

A partire dalla versione 2, poiché le modifiche non sono state più sostanziali come quelle tra la versione 1 e la versione 2, questo framework è diventato noto semplicemente come **Angular**, senza più specifiche del numero di versione.

Appendice C

Il successo di AWS



Figura C.1: Logo di AWS

Al giorno d'oggi il fenomeno del **Cloud computing** è innegabilmente in continua crescita. Con questo termine si fa riferimento a un vero e proprio paradigma di erogazione di risorse informatiche. Tali risorse possono essere dei software ad alte prestazioni veri e propri utilizzabili come dei servizi (da qui il termine di riferimento *SaaS*, ovvero Software as a service) che consentono a chiunque di effettuare archiviazione, elaborazione e trasmissione di dati attraverso la rete internet in modo estremamente efficiente. Questo paradigma è quindi utilizzato dai singoli sviluppatori, ma anche (e soprattutto) dalle tantissime aziende che hanno sempre di più la necessità di garantire ai propri clienti un accesso ai dati rapido da qualsiasi parte del mondo nel modo più economico possibile.

Il mondo del cloud computing non ha sempre avuto tutto questo successo (esso è infatti relativamente recente). Questo perchè le aziende dominatrici del mercato hanno dovuto riorganizzare una parte sostanziale del loro modello di business per abbracciare questo nuovo paradigma (accettandone ovviamente tutti i rischi, primo fra tutti quello legato alla tutela della privacy e dello spionaggio industriale). Col passare del tempo però (visto l'aumento progressivo sia delle richieste dei clienti che dell'aumentare delle moli di dati) quasi tutti si sono resi conto che per garantire ai propri clienti servizi altamente performanti ad un prezzo ragionevole il paradigma che doveva essere adottato era proprio quello del cloud computing.

Tra i grandi colossi dell'informatica (Microsoft e Google per citarne alcuni) appare molto curioso che oggi il leader indiscusso nel mondo del cloud computing sia **Amazon** (società nata come sito web e-commerce e diventata in seguito la più grande *Internet company* al mondo). Essa, con i suoi **Amazon Web Services** (noti al pubblico come AWS) vanta oggi un fatturato di oltre 14 miliardi di dollari (con una crescita annua

stimata attorno al 43%).

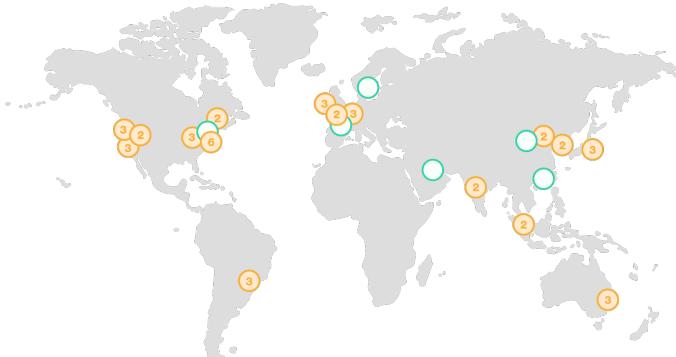


Figura C.2: Infrastruttura globale di AWS

AWS offre effettivamente moltissimi servizi anche abbastanza complessi: essi, per esempio, possono essere piattaforme ricche di tools per lo sviluppo di software performanti, o applicazioni in grado di accedere a grosse moli di dati contenuti in database distribuiti in giro per il mondo ad elevate prestazioni. Oggi dunque ci sono milioni e milioni di clienti che usufruiscono dell'infrastruttura di AWS ogni giorno, anche per far girare il proprio sito web.

Le alte prestazioni di questi servizi sono garantite grazie ad una infrastruttura globale ben studiata: al giorno d'oggi AWS opera in ben 44 zone che sono distribuite su 16 regioni geografiche distinte dislocate in tutto il mondo (ogni regione geografica consta di alcuni *data center* altamente performanti e con una elevata tolleranza ai guasti grazie a connettività ridondanti).

Ciò che però sta sicuramente alla base del successo di AWS di oggi è la politica commerciale innovativa che è stata adottata: è stato infatti usato un approccio chiamato *pay per use* che è risultato conveniente per qualsiasi utilizzatore dei servizi messi a disposizione. Un cliente di AWS paga i servizi soltanto in base al tempo dell'effettivo utilizzo che ne fa, senza alcun vincolo contrattuale o costo fisso di abbonamento. AWS inoltre mette a disposizione diversi piani gratuiti per un tempo limitato che sono sia molto utili per tutti gli sviluppatori che non hanno grandi esigenze sia per le aziende che intendono utilizzare tali servizi per poco tempo (in questo modo i servizi si fanno comunque conoscere, e in futuro le stesse aziende che hanno usufruito dei servizi per un tempo limitato, una volta constatati i vantaggi, potrebbero decidere di usufruire del servizio completo con i costi che ne conseguono).

I servizi di AWS sono davvero molti. Tra i più famosi troviamo sicuramente **DynamoDB** (database ad elevatissime prestazioni che può ridurre i tempi di risposta da millisecondi a microsecondi, addirittura anche quando le richieste superano il milione al secondo) o il servizio **AWS Lambda**, che permette direttamente al singolo sviluppatore di scrivere codice in un certo linguaggio di programmazione, volto all'elaborazione dei dati remoti (tipicamente il codice viene eseguito in base allo scatenarsi di eventi) senza alcuna preoccupazione riguardo la gestione dei server (il prezzo del servizio sarà

calcolato in base al tempo di elaborazione del codice); o ancora il servizio **CloudFront**, che permette di erogare i dati da tutte le regioni di AWS contemporaneamente (in questo modo al cliente sarà sempre garantito il massimo delle performance, in qualsiasi parte del mondo).

Per finire, AWS sta facendo seri progressi anche in moltissimi altri ambiti, per esempio quello dell'*Intelligenza Artificiale*. Molte delle funzionalità sono infatti basate su algoritmi di apprendimento automatico, che consentono agli utilizzatori dei servizi l'uso di funzionalità intelligenti come quelle basate sul riconoscimento vocale automatizzato. Altro progresso importante da menzionare è quello in ambito della *Sicurezza* e della *Verifica del software*: grazie infatti a diversi *tools* di analisi statica per la verifica formale del software sviluppati internamente a AWS è stato possibile constatare che i servizi di AWS contribuiscono a garantire servizi sicuri e robusti^[6].

Bibliografia

Siti web di riferimento:

- [1] Sito ufficiale di IKS: <https://www.iks.it>
- [2] Sito ufficiale di AWS: <https://aws.amazon.com>
- [3] Sito ufficiale di Angular: <https://angular.io>
- [4] Sito ufficiale del linguaggio TypeScript: <http://www.typescriptlang.org>
- [5] “Ok... let me explain: it’s going to be Angular 4.0, or just Angular”, *Igor Minar*:
<http://angularjs.blogspot.it/2016/12/ok-let-me-explain-its-going-to-be.html>
- [6] “Automated formal reasoning about amazon web services”, *Byron Cook*:
<https://dl.acm.org/citation.cfm?doid=3092282.3092315>

Altri siti web di riferimento:

- * Wikipedia: <https://www.wikipedia.org>

Libri consultati:

- * UML e Design Pattern, *Massimiliano Bigatti*, HOEPLI Informatica
- * Design Patterns, *Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides*, Pearson

Slide dei corsi dell’UNIPD:

- * Slide del corso di Ingegneria del software del prof. Tullio Vardanega: <http://www.math.unipd.it/~tullio/IS-1/2016>

Glossario

A

API

Acronimo di Application Programming Interface (ovvero interfaccia di programmazione di una applicazione) si indicano un insieme di procedure (raggruppate assieme per formare un insieme di strumenti specifici) per il compimento di un determinato compito all'interno di un certo programma.

B

Black-box testing

Si definisce di tipo Black-box un test di accertamento delle funzionalità di un software effettuato solamente tramite l'interfaccia utente, senza osservare direttamente i suoi meccanismi interni.

Breadcrumb

Rappresenta una tecnica di navigazione usata nelle interfacce utente delle pagine web. Forniscono agli utenti il modo di tenere traccia della loro posizione attuale in relazione all'intera struttura di un sito (o di una applicazione) web. Solitamente appaiono orizzontalmente nella parte superiore della pagina.

Business Critical System

Tipologie di sistemi la cui correttezza e integrità sono essenziali per la buona riuscita di un'operazione legata a impresa o mercato.

D

Debugger

Programma progettato per l'individuazione statica di bug all'interno di un codice sorgente.

Dependency Injection

Nome di un design pattern particolare della programmazione orientata agli oggetti che consiste nella risoluzione delle dipendenze di una specifica classe verso un'altra classe. La classe che dipende dall'altra, dichiara una dipendenza verso di essa (tipicamente in modo esplicito all'interno del proprio costruttore), ma quando la classe sarà istanziata sarà un'altra componente, chiamata *container*, a risolvere la dipendenza dichiarata in questo modo: il *container* istanzierà la classe dipendente, la salverà in un contenitore di istanze e la ritornerà ogniqualvolta ce ne sarà il bisogno.

Design Pattern

Questo termine fa riferimento a una soluzione progettuale generale ad un problema ricorrente che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software.

H

Hosting

Termine con il quale si indica un particolare servizio di rete che alloca su un server delle risorse di un certo tipo, rendendole così accessibili dalla rete Internet ai suoi utenti.

J

JSON

Acronimo di JavaScript Object Notation, è un formato dichiarativo basato sullo Standard ECMA-262 adatto all'interscambio di dati fra applicazioni client-server.

N

NoSQL

Questo termine fa riferimento a sistemi software dove la persistenza dei dati è caratterizzata dal fatto di non utilizzare il modello relazionale. L'espressione NoSQL fa riferimento al linguaggio SQL, linguaggio più usato nell'ambito di interrogazione dei dati nei database relazionali.

P

Provisioning

Processo mediante il quale un amministratore di sistema assegna risorse e privilegi, non solo agli utenti di una rete ma anche a chi le utilizza da remoto.

Q

QR-CODE

Codice a barre bidimensionale in forma matriciale utilizzato per memorizzare informazioni generalmente destinate a essere lette tramite un dispositivo mobile come uno smartphone.

R

REST

Acronimo di REpresentational State Transfer, con questo termine si fa riferimento a un tipo di architettura software per i sistemi di ipertesto (come il web), usata tipicamente insieme al protocollo HTTP.

S

SDK

Acronimo di Software Development Kit, si riferisce genericamente a un insieme di strumenti per migliorare lo sviluppo o la documentazione di software.

Single Responsibility Principle

Principio (facente parte della famiglia dei principi *SOLID*) della programmazione ad oggetti che afferma che ogni elemento di un programma (ovvero una classe, un metodo, o una variabile) deve avere una e una sola responsabilità (i servizi offerti da questo elemento dovrebbero quindi essere strettamente allineati a tale responsabilità).

Singleton

Nome di un design pattern che consiste nella creazione di una classe all'interno di una architettura, che però sarà disponibile alle altre componenti solo ed esclusivamente come una singola istanza.

Storage

Termine che fa riferimento a dispositivi hardware, infrastrutture e supporti per la memorizzazione non volatile di grandi quantità di informazioni in formato elettronico.

Ringraziamenti

Seduto davanti a questo PC a scrivere le ultime righe, ancora non mi capacito che il momento di gloria sia finalmente arrivato.

Il pensiero in assoluto più importante di tutti va a mia mamma Palma, per avermi permesso, grazie al suo sostegno, di vivere una esperienza fantastica e difficile da raccontare, oltre che per aver sempre creduto in me e nelle mie capacità senza la pretesa di avere nulla in cambio. Un pensiero importantissimo va anche a mia zia Lavinia e a mio nonno Vincenzo, che è stato per me più come un padre che come un nonno.

Vorrei ringraziare caldamente il prof. Francesco Ranzato, relatore della mia tesi di laurea, per il sostegno fornитomi durante la stesura della stessa, e il prof. Massimo Marchiori, che grazie al suo carisma è riuscito come nessun altro a farmi appassionare a questo ambito.

Ringrazio Tobia Zorzan, mio tutor aziendale, per la cortesia, la professionalità dimostrata e per tutto il tempo dedicatomi in azienda durante l'intero percorso di questo stage.

Un ringraziamento va anche ai colleghi del corso di Informatica e al gruppo *Visions Team*, per aver condiviso fatiche e soddisfazioni, e per aver reso la giornata del 15 maggio come uno dei giorni più dimenticabili della mia vita.

Ringrazio di cuore tutti i ragazzi conosciuti nella Residenza Messori, in particolar modo Giuseppe, Simon, Giuliano, Chiara, Alessio e Giacomo che per me sono ormai diventati praticamente dei familiari. Senza la Residenza non sarei quello che sono ora e sono sicuro di poter dire che ricorderò per sempre questo periodo come uno dei più belli ed emozionanti in assoluto di tutta la mia vita.

Un ringraziamento particolare va anche al prof. Massimo Rea e al dr. Damiano Donadello, nei quali ho sempre trovato un punto di appoggio nelle situazioni più delicate.

Infine, un caloroso ringraziamento a me stesso, unica persona al mondo davvero in grado di sopportarmi.

Padova, Dicembre 2017

Claudio Zanacchi