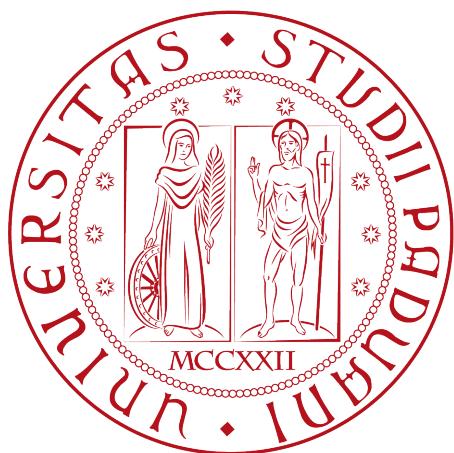


Università degli Studi di Padova
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"
CORSO DI LAUREA IN INFORMATICA



**Protocolli di sicurezza in ambito web per
l'applicazione “SyncTrace” di prevenzione
Covid-19**

Tesi di laurea triennale

Relatore

Prof. Tullio Vardanega

Laureando

Mariano Sciacco

ANNO ACCADEMICO 2019-2020

Mariano Sciacco: *Protocolli di sicurezza in ambito web per l'applicazione “SyncTrace” di prevenzione Covid-19*, Tesi di laurea triennale, © Luglio 2020.

Sommario

Il presente documento descrive dettagliatamente il tirocinio da me svolto presso l'azienda Sync Lab s.r.l. nella sede di Padova durante il periodo che va dal 04-05-2020 al 26-06-2020.

L'esperienza di stage presso l'azienda Sync Lab ha avuto una durata complessiva di 300 ore ed è stata supervisionata e coordinata sia dal mio *tutor* aziendale, l'ingegnere Fabio Pallaro, che dal mio relatore presso l'ateneo, prof. Tullio Vardanega.

Lo scopo dello stage era di analizzare i principali protocolli di sicurezza utilizzati in ambito web, valutandone gli aspetti positivi e negativi, identificando poi la soluzione progettuale più adatta, seguita dallo sviluppo di una libreria che fosse integrabile con l'applicativo web del progetto *SyncTrace* per la prevenzione della malattia COVID-19, diffusasi a inizio 2020.

Il percorso di tirocinio ha richiesto lo studio di nuove tecnologie in ambito di autorizzazione e autenticazione web e ha previsto un periodo di formazione anche su nuovi *framework* (*Angular*) e linguaggi di programmazione (*TypeScript*, *Javascript*).

Ho suddiviso il presente documento in 4 capitoli:

- **Capitolo 1:** presentazione del contesto organizzativo e produttivo aziendale, con approfondimento sui processi aziendali interni e sullo spirito di innovazione;
- **Capitolo 2:** presentazione dell'offerta di stage con gli obiettivi, i vincoli e le motivazioni che mi hanno spinto a scegliere l'azienda;
- **Capitolo 3:** presentazione dettagliata del progetto con particolare approfondimento delle fasi del progetto, delle tecnologie trattate e delle soluzioni progettuali attuate per la realizzazione dei prodotti attesi dall'azienda;
- **Capitolo 4:** resoconto conclusivo con una valutazione retrospettiva del percorso di tirocinio, riguardante gli obiettivi raggiunti, le difficoltà e le competenze professionali maturate.

“In an extreme view, the world can be seen as only connections, nothing else. We think of a dictionary as the repository of meaning, but it defines words only in terms of other words. I liked the idea that a piece of information is really defined only by what it’s related to, and how it’s related. There really is little else to meaning. The structure is everything. There are billions of neurons in our brains, but what are neurons? Just cells. The brain has no knowledge until connections are made between neurons. All that we know, all that we are, comes from the way our neurons are connected.”

— Tim Berners-Lee, *Weaving the Web* (1999)

Ringraziamenti

 Lorem ipsum sit dolor amet

Padova, Luglio 2020

 Mariano Sciacco

Indice

1 Analisi del contesto aziendale	1
1.1 Introduzione	1
1.2 L'azienda Sync Lab	2
1.3 Organizzazione del lavoro	3
1.3.1 Il modello di sviluppo agile	4
1.4 Tecnologie di interesse	6
1.5 Prodotti e innovazione	9
2 Lo stage in Sync Lab	13
2.1 Lo stage per l'azienda	13
2.2 Il contesto di attualità	14
2.3 La risposta dell'azienda	15
2.4 Dal problema alla visione del prodotto	16
2.5 Lo scopo dello stage	19
2.6 Motivazione della scelta	20
2.7 Vincoli e obiettivi dello stage	21
2.7.1 Pianificazione temporale	22
2.7.2 Vincoli organizzativi	23
2.7.3 Obiettivi e prodotti attesi	26
3 Il progetto di stage	29
3.1 Organizzazione e configurazione	29
3.1.1 Automazione del workflow	33
3.2 Analisi dei protocolli di sicurezza	35
3.2.1 Protocollo OAuth 2.0	35
3.2.2 Protocollo OpenID Connect	40
3.2.3 JSON Web Token	41
3.2.4 Principali vulnerabilità di OAuth 2.0	43
3.2.5 Applicabilità dei protocolli di sicurezza web	45
3.3 Proof of Concept	47
3.4 Progettazione	51
3.4.1 Identificazione delle tecnologie	51
3.4.2 Requisiti della libreria	52
3.4.3 Soluzioni progettuali	55
3.4.4 Architettura della libreria	57
3.5 Codifica e verifica del software	60
3.5.1 Verifica della libreria	63
3.6 Validazione e collaudo	66

3.7	Integrazione del prodotto in SyncTrace	67
4	Resoconto conclusivo	71
4.1	Raggiungimento degli obiettivi	71
4.1.1	Prodotti realizzati	72
4.1.2	Aspettative soddisfatte	72
4.2	Conoscenze professionali maturate	73
4.3	Valutazione personale	74

Elenco delle figure

1.1 Rappresentazione grafica del virus SARS-CoV-2	1
1.2 Motto aziendale e punti di forza di Sync Lab	2
1.3 Sedi di Sync Lab in Italia	3
1.4 Alcuni degli ambiti specialistici di Sync Lab	4
1.5 Metodo Scrum	5
1.6 Organizzazione delle attività nella Scrum Board	6
1.7 Design architettonale MVC di AngularJS	7
1.8 Estratto di una bacheca di Trello	8
1.9 Homepage di SynClinic	9
1.10 Alcuni dei progetti di ricerca e sviluppo di Sync Lab	10
2.1 P.d.C. Giuseppe Conte durante la pandemia	14
2.2 Tracciamento dei contatti tramite smartphone	15
2.3 Dati sul contact tracing nell'applicazione SyncTrace	17
2.4 Diagramma dei componenti per l'applicazione SyncTrace	18
2.5 Funzionamento ad alto livello di OAuth 2.0	19
2.6 Pagina del sito Covid19-Italy per la variazione giornaliera dei dati	21
2.7 Estratto del diagramma di Gantt per la ripartizione delle attività	23
2.8 Estratto del registro delle attività giornaliere di stage	24
2.9 Piattaforma web di GitLab	25
2.10 Efficienza ed efficacia in una vignetta	28
3.1 Software Stoplight Studio per la definizione delle API	29
3.2 Software GitKraken per le operazioni sulla repository di progetto	30
3.3 Esempio di chiamata HTTP con politica CORS	32
3.4 Esecuzioni della CI nella repository di progetto	33
3.5 Logs della CI nella repository di progetto	35
3.6 Una funzionalità del sito web Yelp	36
3.7 Autorizzazione OAuth 2.0 su Yelp	36
3.8 OAuth 2.0, Code flow nel dettaglio	37
3.9 Diagramma di sequenza per l'accesso a una risorsa protetta	39
3.10 OAuth 2.0, dettaglio Implicit flow	39
3.11 OpenID Connect, Code flow	40
3.12 Diagramma di sequenza di per l'accesso con il protocollo OpenID Connect	41
3.13 Struttura di un JWT	42
3.14 Consenso OAuth su Google nel 2017	45
3.15 Proof of concept, pagina protetta che sfrutta il protocollo OAuth 2.0 .	48

3.16 Proof of concept, errore per mancata autorizzazione sfruttando il protocollo OAuth 2.0	49
3.17 Proof of concept, pagina del client che vuole eseguire l'accesso alla web app usando OpenID Connect	49
3.18 Proof of concept, servizio di identità esterno per l'accesso alla web app tramite OpenID Connect	50
3.19 Proof of concept, redirect e accesso finale all'applicazione web usando OpenID Connect	50
3.20 Diagramma per il design pattern Facade	57
3.21 Diagramma dei package della libreria di progetto	58
3.22 Diagramma di sequenza per il processo di autenticazione utilizzando la libreria di progetto	59
3.23 Diagramma di sequenza per la richiesta di una risorsa via HTTP utilizzando un servizio della libreria di progetto	59
3.24 Pagina login dell'applicazione di esempio per mostrare il funzionamento della libreria	60
3.25 Pagina protetta dell'applicazione di esempio per mostrare il funzionamento della libreria	61
3.26 Disposizione delle cartelle nell'applicazione di esempio del progetto Angular	62
3.27 Esecuzione di Jasmine nel browser	63
3.28 Numero di casi di test realizzati nell'attività di verifica	65
3.29 Code coverage nell'attività di verifica	65
3.30 Grafico riassuntivo dei requisiti soddisfatti nell'attività di validazione	66
3.31 Esempio di albero dei contatti dell'applicativo web di SyncTrace	68
3.32 Pagina di accesso dell'applicativo web di SyncTrace	69
4.1 Grafico riassuntivo del code coverage raggiunto in base alle soglie	72

Elenco delle tabelle

2.1 Tabella riassuntiva delle ore settimanali di stage	22
2.2 Tabella riassuntiva degli obiettivi di stage	26
2.3 Tabella riassuntiva delle soglie di Code Coverage del prodotto	28
3.1 Tabella riassuntiva delle asserzioni principali di JWT	43
3.2 Tabella riassuntiva sulla scelta dei servizi da implementare nella libreria	52
3.3 Tabella riassuntiva dei casi d'uso identificati per il progetto	53
3.4 Tabella riassuntiva dei requisiti identificati per il progetto	54
3.5 Estratto di alcuni requisiti funzionali del progetto	54

ELENCO DELLE TABELLE

xi

4.1	Tabella riassuntiva degli obiettivi di stage raggiunti	71
4.2	Tabella riassuntiva dei prodotti di stage completati	72

Capitolo 1

Analisi del contesto aziendale

1.1 Introduzione

L'azienda in cui ho svolto lo stage è Sync Lab, un'azienda che si è molto affermata in Italia nel settore dell'*Information and Communication Technology* (ICT) e che mi ha permesso di svolgere uno stage con una tematica nuova e molto attuale, visto il decorso degli eventi avvenuti a inizio febbraio 2020.

Lo scoppio della pandemia *COVID-19* ha creato diverse difficoltà organizzative, sia per me che per l'azienda, ma con l'adozione dello *smart working* ho potuto comunque avviare l'attività lavorativa remotamente da casa, gestendo in modo più efficace il tempo a disposizione. In ottica di prevenzione, l'accesso alla sede aziendale è stato inizialmente negato e solo nelle ultime settimane di stage ho potuto accedervi, seguendo le normative per ridurre il rischio di contagio.

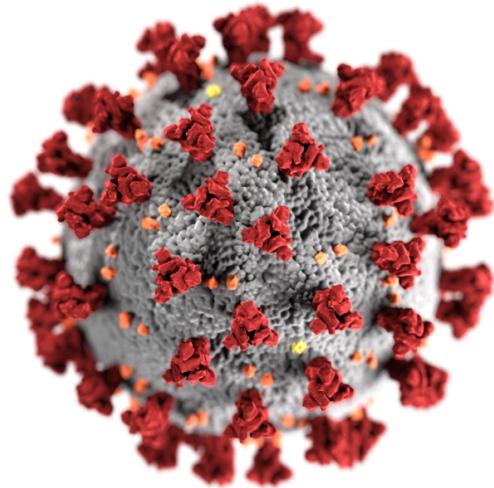


Figura 1.1: Rappresentazione grafica del virus SARS-CoV-2
Fonte: it.wikipedia.org

La mia esperienza di stage non è stata minata completamente da questi avvenimenti e, grazie alla presenza in sede nelle ultime settimane, ho potuto comunque raccogliere maggiori informazioni sull'organizzazione e sui settori di interesse dell'azienda. Pertanto,

le informazioni riportate in questo capitolo sono frutto della mia esperienza in buona parte remota e in piccola parte in presenza. Ho verificato l'attendibilità di queste informazioni sia con il materiale accessibile pubblicamente (es. sito web ufficiale), sia con le interazioni avvenute con il *tutor* aziendale, che mi ha illustrato con maggiore rilievo i settori, le tecnologie e i prodotti di interesse per l'azienda.

1.2 L'azienda Sync Lab

Sync Lab è un'azienda italiana nata nel 2002 con sede principale a Napoli, che si è affermata rapidamente nel mercato dell'*Information and Communication Technology* (ICT), collaborando con diversi clienti per offrire progettazione, realizzazione e manutenzione di soluzioni IT.

L'azienda ha puntato molto sull'innovazione e, dopo essere partita come *software house*, è diventata *system integrator*, maturando molto dal punto di vista tecnico-metodologico.



Figura 1.2: Motto aziendale e punti di forza di Sync Lab
Fonte: synclab.it

L'azienda ha quindi cominciato a operare anche in nuovi settori, quali la *cybersecurity*, la videosorveglianza e il *mobile*, promuovendo così una buona attività di ricerca e sviluppo. Il numero di dipendenti conta oltre 250 risorse, dislocate nelle 5 sedi sul territorio italiano: Roma, Napoli, Milano, Verona e Padova.



Figura 1.3: Sedi di Sync Lab in Italia

Fonte: synclab.it

Tutte le sedi collaborano tra di loro con l'ausilio di molti specialisti per supportare i propri clienti e propri partner per diverse soluzioni aziendali.

Alcuni tra i clienti più rilevanti sono: *Enel, Sky, Vodafone, Tim, Fastweb, Trenitalia, Poste Italiane, UniCredit e Intesa SanPaolo*.

1.3 Organizzazione del lavoro

Per quanto riguarda l'organizzazione del lavoro, l'approccio di Sync Lab si basa sulla conoscenza dei processi che si fondano sul modello di sviluppo Agile. Tale modello è stato caratterizzante anche nello stage per tutta la sua durata, i cui obiettivi seguiti dall'azienda si possono riassumere in tre punti fondamentali:

- **comprendere il contesto operativo del cliente** col fine da delineare le necessità specifiche di ognuno;
- porsi l'obiettivo di **fornire al cliente un supporto mirato** con apposite indicazioni organizzative, operative e di processo;
- a partire dai processi interni, **accelerare e favorire la formazione di soluzioni** all'interno della struttura del cliente.

Sulla base di questi principi, l'azienda conta di raggiungere i propri obiettivi mettendo in atto i seguenti processi:

- **Consulenza:** attraverso la collaborazione con specialisti del settore, l'azienda vuole fornire una consulenza ai suoi clienti che porti un'evoluzione in termini di competitività, sviluppo e innovazione tecnologica;

- **Fornitura:** l'azienda si mette a disposizione per la realizzazione di un prodotto del cliente e, parallelamente, mette in atto delle attività che migliorino questo processo. In particolare:
 - controllo e ottimizzazione della qualità del software in base al contesto aziendale;
 - verifica delle procedure aziendali con eventuali correzioni;
 - analisi e miglioramento degli standard di qualità aziendali (es. ISO 9001);
- **Sviluppo:** basandosi sul modello di sviluppo agile, Sync Lab mostra al cliente e agli *stakeholders* l'evoluzione dello sviluppo del prodotto richiesto, raccogliendo tutti i *feedback* che possono essere determinanti per la loro soddisfazione;
- **Manutenzione:** a seguito del rilascio di un prodotto, l'azienda svolge delle attività di manutenzione per tutta la vita del software, riadattando nuove funzionalità in base alle esigenze del cliente e applicando correzioni ove necessario. Nello specifico, Sync Lab offre 3 tipologie di manutenzione:
 - **manutenzione correttiva**, per correggere eventuali difetti del prodotto;
 - **manutenzione adattiva**, per riadattare il software a cambiamenti dell'ambiente di produzione e dell'architettura;
 - **manutenzione evolutiva**, per aggiungere ed estendere con nuove funzionalità il prodotto.



Figura 1.4: Alcuni degli ambiti specialistici di Sync Lab
Fonte: synclab.it

1.3.1 Il modello di sviluppo agile

L'azienda adotta un modello di sviluppo agile che pone le proprie radici nel metodo *Scrum*. Gli *stakeholders*, infatti, vengono costantemente coinvolti nel processo di sviluppo del prodotto per raccogliere *feedback*, che sono fondamentali per la buona riuscita del progetto.

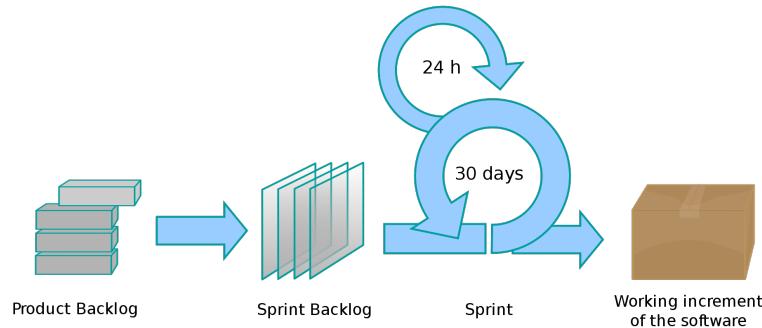


Figura 1.5: Metodo Scrum
Fonte: it.wikipedia.org

Basandomi sull'esperienza di stage, le fasi del progetto, che prendono il nome di *sprint*, sono organizzate in modo flessibile e adattabile in base alle esigenze. Ad ogni *sprint* corrisponde l'introduzione di una nuova funzionalità, che viene opportunamente verificata e comprovata dalla soddisfazione del cliente. Pertanto, il *modus operandi* si articola come segue:

1. definisco un ***product backlog***, dove riporto la lista delle cose da fare ("to-do") in una *scrum board*, con tutte le attività e i requisiti posti dal progetto;
2. realizzo uno ***sprint planning***, per pianificare preventivamente gli *sprint* che ho intenzione di attuare;
3. definisco uno ***sprint backlog***, ossia un sottoinsieme di obiettivi da raggiungere durante un singolo *sprint* sulla base del mio *product backlog*;
4. eseguo lo ***sprint*** in un lasso di tempo limitato, 1-2 settimane nel mio caso, massimo 30 giorni generalmente;
5. revisiono lo ***sprint goal***, in cui determino l'incremento effettivo al termine di uno *sprint*.

Durante lo sviluppo, l'azienda organizza anche dei meeting insieme agli *stakeholders* e ai membri del progetto per valutare frequentemente l'andamento degli *sprint*. In particolare, ho avuto modo di partecipare alle seguenti riunioni, che fanno fede al metodo *Scrum*:

- ***daily scrum***: riunione giornaliera di breve durata con i membri del team per trattare l'attuale andamento dello *sprint*, rispondendo a tre domande:
 - cosa è stato fatto fino ad oggi?
 - cosa si farà per domani?
 - quali impedimenti rallentano o bloccano i miei progressi?
- ***sprint review***: riunione di fine *sprint* in cui si ispezionano i risultati dell'incremento insieme agli *stakeholders*, decidendo eventuali cambiamenti da apportare al *product backlog*;

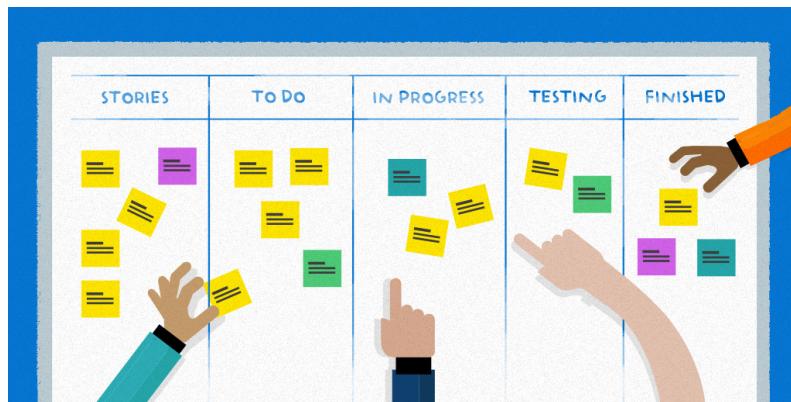


Figura 1.6: Organizzazione delle attività nella Scrum Board

Fonte: govwebworks.com

Dopo aver parlato con il mio *tutor* aziendale, l'ingegnere Fabio Pallaro, posso sicuramente affermare che il modello *Scrum* applicato nell'ottica di *Smart working* funziona adeguatamente. Infatti, gran parte dei *meeting* a cui ho partecipato si sono svolti con piena regolarità in videoconferenza e la pianificazione degli *sprint* non ha sollevato grosse criticità. La gestione delle attività attraverso la *Scrum board* può essere eseguita sfruttando degli appositi strumenti online (es. *Trello*), in cui riporto lo stato di avanzamento di ogni specifico *sprint*. Il tempo a disposizione, infine, viene organizzato meglio, tolto i tempi di percorrenza per giungere alla sede e la comunicazione diventa più mirata quando si utilizzano servizi di messaggistica istantanea (es. *Discord*, *Slack*).

1.4 Tecnologie di interesse

Come menzionato precedentemente, Sync Lab offre un gran numero di servizi in molti campi del settore IT (*Information technology*) e fa uso di molte tecnologie per realizzare dei prodotti solidi. L'azienda ha scelto di operare con molti linguaggi in base alle soluzioni richieste e alcuni di questi sono i seguenti:

- **Java:** un linguaggio a oggetti molto famoso e ampiamente usato negli applicativi di Sync Lab, specialmente con l'integrazione del framework *Spring*, sfruttato principalmente per lo sviluppo di servizi *REST* in ambito web;
- **Javascript:** un linguaggio a eventi utilizzato per la realizzazione di applicativi web lato *client*, spesso impiegato per l'implementazione di effetti interattivi nel sito web;
- **TypeScript:** un linguaggio a oggetti utilizzato molto nel framework *Angular*, utile per realizzare i servizi e la "logica" delle *Single Page Application* in ambito web;
- **HTML5 e CSS:** linguaggi di *markup* e di stile comunemente usati per la realizzazione dello scheletro e della veste grafica dei siti web;
- **Kotlin:** un linguaggio di programmazione multi-paradigma, usato dall'azienda per lo sviluppo delle applicazioni *Android*.

Per quanto riguarda i *framework*, Sync Lab ha scelto di adottare soluzioni che fossero mature, solide e facilmente mantenibili nel corso del tempo. In particolare, alcuni dei *framework* usati dall'azienda sono:

- **Spring**: *framework* basato su *Java*, che viene utilizzato per la realizzazione di applicativi lato *server*, quali servizi *REST*, ma anche applicazioni *desktop* e web. Spring mette a disposizione delle funzionalità che promuovono l'utilizzo dei *design pattern*, specialmente per la realizzazione di un'architettura che si mantenga solida nel tempo ed estendibile in futuro;
- **Angular e AngularJS**: *framework* utilizzati nell'ambito web per lo sviluppo di *Single Page Application* che siano veloci e reattive, appoggiandosi a un *backend* con servizi *REST*. Angular sfrutta il motore di *javascript* e permette di ottenere un'esperienza nella navigazione di un sito che è pari a quella di una applicazione *desktop*.

Nelle ultime versioni, *Angular* ha introdotto alcune funzionalità relative alla gestione delle dipendenze e dei test che lo rendono ancora più semplice da apprendere e da mantenere. L'utilizzo del pattern *Model-View-ViewModel* (e *Model-View-Controller* in *AngularJS*), inoltre, rende facilmente estensibili gli applicativi a livello di codice e promuove un'ottima *best practice* nell'uso dei *design pattern*.

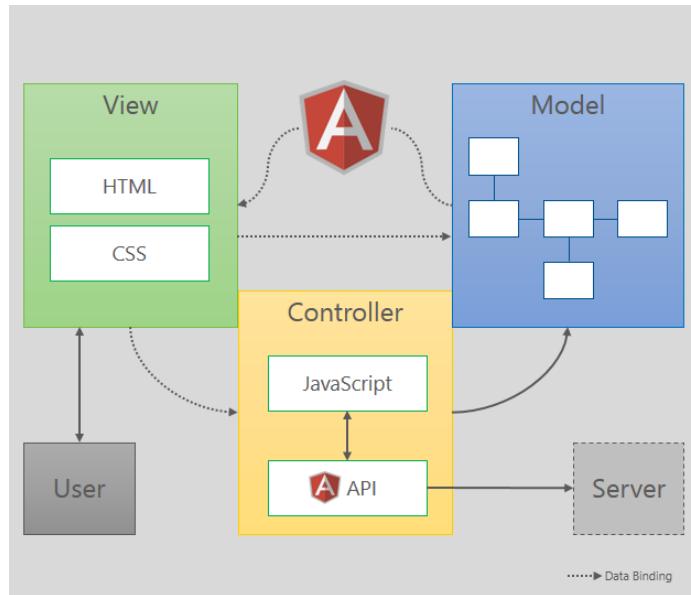


Figura 1.7: Design architettonico MVC di AngularJS
Fonte: software-architects.com

Nel periodo della pandemia, l'azienda ha introdotto diverse soluzioni di tipo organizzativo col fine di migliorare l'interazione e la comunicazione dei gruppi di lavoro. Lo *smart working* ha portato l'azienda a provare nuove soluzioni che potessero essere facilmente adattabili alle necessità dei dipendenti e dei gruppi di progetto, talvolta migliorando l'efficienza nei processi aziendali. Per questi motivi, anche nel mio stage l'azienda ha utilizzato i seguenti strumenti:

- **Google Meet:** software utilizzato per le videoconferenze che permette di comunicare con più colleghi aziendali e organizzare riunioni virtuali con un grande numero di partecipanti, sincronizzando la propria agenda con il resto della *suite* di Google (es. *Google Calendar*);
- **Discord:** *social network* gratuito in cui è possibile fruire di *chat* in tempo reale, gestendo più canali vocali e testuali divisi per argomento. *Discord* può essere usato anche su più sistemi operativi, sia desktop che mobili e, rispetto ad altre soluzioni come *Slack*, è completamente gratuito nelle funzionalità principali. Tra le varie funzioni disponibili, ad esempio, permette di dividere i membri in **gruppi**, per identificare meglio ciascuno con il proprio progetto e i propri permessi. La parte di videoconferenze, tuttavia, è meno avanzata rispetto a *Google Meet*, dato che tra le tante mancanze non permette di "programmare" le riunioni;
- **Trello (Scrum Board):** nell'ambito del metodo *Scrum*, *Trello* risulta essere un'ottima soluzione per gestire e organizzare il proprio *workflow* online, costituendo a tutti gli effetti una *Scrum Board*. Attraverso *Trello*, infatti, è possibile rappresentare le attività e gli *sprint* in **schede** (simili ai "post-it" in una *Scrum Board*), le quali vengono inserite sotto apposite colonne di avanzamento. Per ciascuna scheda è possibile spostare manualmente o automaticamente lo stato di avanzamento in base a quanto realmente realizzato. Le diciture delle colonne sono generalmente le seguenti:
 - **Backlog**, che contiene le schede di requisiti e attività del progetto, definiti nel *product backlog*;
 - **Da fare**, ossia gli *sprint* e le attività da realizzare;
 - **In corso**, ossia gli *sprint* e le attività in esecuzione;
 - **Da verificare**, ossia gli *sprint* e le attività concluse e in attesa di verifica (*sprint review*);
 - **Verificati**, ossia gli *sprint* e le attività concluse e verificate.

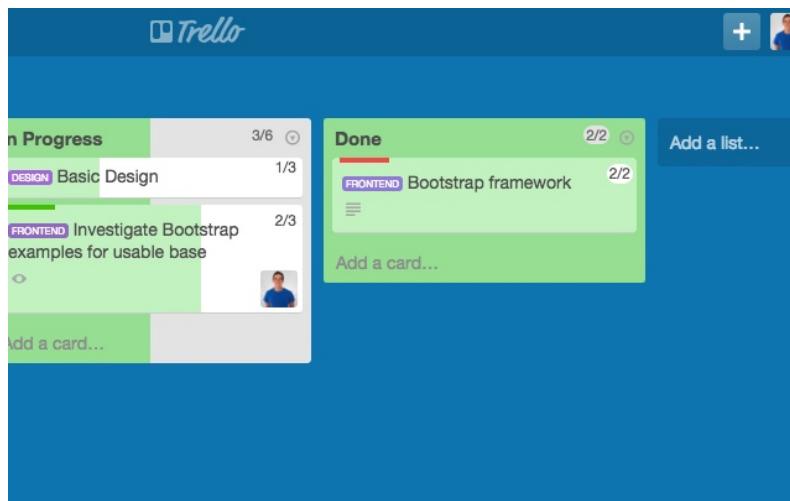


Figura 1.8: Estratto di una bacheca di Trello

Fonte: chrome.google.com

1.5 Prodotti e innovazione

Sync Lab lavora in molti ambiti del settore informatico, quali *business consultancy*, *IT consultancy* e *Project Financing*, e nel corso degli anni ha sviluppato un gran numero di prodotti sottolineando l'importanza della qualità come garanzia. Per questo motivo l'azienda ha ottenuto molte certificazioni, tra cui ISO 9001, ISO 14001, ISO 27001 e ISO 45001.

Nell'ambito della ricerca e sviluppo, l'azienda vanta di un buon numero di prodotti caratterizzati dall'innovazione e dalla qualità. Alcuni di questi prodotti sono i seguenti:

- **SynClinic:** software integrato per la gestione delle strutture sanitarie con funzionalità di cartella clinica, fatturazione e magazzino, che permette di gestire, organizzare e monitorare tutte le fasi del percorso di cura del paziente, indispensabile quindi per il personale sanitario a fronte del rischio clinico. Il prodotto è stato realizzato implementando l'architettura a microservizi su linguaggio *Java* e un *frontend* scritto in *typescript* con l'utilizzo del *framework Angular*;



Figura 1.9: Homepage di SynClinic
Fonte: synclinic.it

- **DPS 4.0:** software web per la gestione del *General Data Protection Regulation (GDPR)* in più aziende con una soluzione guidata per aggiornare e modificare i documenti di *privacy* in modo conforme agli standard di riferimento. La parte *backend* di *DPS 4.0* è stata realizzata completamente con *Java* (*framework Spring*), mentre per la gestione a eventi è stato utilizzato come *broker* il prodotto *RabbitMq* e come *frontend* il *framework Angular*;
- **StreamLog:** software di gestione della *compliance* al provvedimento del Garante per la protezione dei dati personali relativo agli Amministratori di Sistema. In particolare, permette di finalizzare il soddisfacimento dei requisiti fissati dal Garante, effettuando controllo degli accessi degli utenti in modo semplice ed efficace. A livello di sviluppo, è stato utilizzato un pattern architettonicale di monolite unico in cui converge sia la parte *backend*, che quella *frontend*;
- **StreamCrusher:** rappresenta una tecnologia per finalizzare il supporto all'informazioni in merito a decisioni di business, evidenziando velocemente criticità

e riorganizzando i processi in base alle nuove esigenze. Il motore (*core*) del prodotto è stato implementato unicamente su linguaggio *Java standard*;

- **Magie**: soluzione di monitoraggio applicativo in *real-time* in grado di raccogliere dati eterogenei, aggregarli ed elaborarli al fine di rappresentarli su una dashboard di *business intelligence*. Il prodotto è stata realizzato a partire dallo *stack ELK* (*Elastic Search, Logstash e Kibana*);
- **FidCity**: soluzione software di *proximity marketing* integrata con componente social su *web app* e *app mobile*. FidCity è stato sviluppato interamente sia su piattaforma *Android*, che su *iOS*, come app native.
- **Wave**: acronimo di *Wide Area Videosurveillance Environment*, viene utilizzato nell'ambito dei sistemi di videosorveglianza per integrare i Sistemi Informativi Territoriali (GIS), abilitando un controllo totale dell'area da sorvegliare; *Wave*, di fatto, è un plugin che può essere installato sulla piattaforma *Milestone System A/S*.

Sync Lab collabora con un grande bacino di clienti, e questo crea la necessità di essere sempre in costante aggiornamento per garantire **innovazione** nelle soluzioni software. Per l'azienda potrei affermare che la parola chiave è **reattività**, dal momento che per inseguire l'innovazione Sync Lab è divisa in 3 dipartimenti principali:

- **Research and Development (R & D)**, dove l'azienda promuove nuovi prodotti eseguendo ricerca e sviluppo in più settori, alimentando così il profilo aziendale e le proprie competenze nel mercato;
- **Lab**, in cui l'azienda mette in atto i risultati derivanti dal dipartimento *R & D*, promuovendo soluzioni che migliorino ed estendano l'innovazione tecnologica;
- **Start-up**, dove l'azienda collabora e promuove le start-up con maggiore successo in termini di innovazione, sia in Italia che all'estero, come nel caso di *SoberEye* e *PromoQui*.



Figura 1.10: Alcuni dei progetti di ricerca e sviluppo di Sync Lab
Fonte: synclab.it

Questi dipartimenti collaborano tra di loro in modo **sincrono** per mettere in atto cambiamenti che siano determinanti sia nelle soluzioni software, che nei processi coinvolti

nello sviluppo e l'organizzazione del lavoro. L'azienda, di recente, ha aperto i propri confini a nuove tematiche negli ambiti di *Cybersecurity*, *E-Health*, *Blockchain* e *Big Data* formando degli appositi **progetti di ricerca** nelle diversi sedi per sperimentare e imparare nuove tecnologie.

Sfruttando le potenzialità di *Discord*, i colleghi dell'azienda si scambiano spesso messaggi nei canali testuali quando si imbattano in soluzioni migliori e innovative, e questo porta a molti argomenti di discussione che, talvolta, mi hanno personalmente coinvolto. Analogamente, anche in presenza mi è spesso capitato di partecipare a discussioni con altri specialisti del settore che mi hanno portato ad aumentare la mie conoscenze e a migliorare nel corso dello sviluppo del mio prodotto.

Questa spinta all'innovazione, pertanto, l'ho ritrovata nello stesso progetto di stage che rispecchia i canoni e i principi dell'azienda nel provare a realizzare qualcosa di nuovo e di diverso rispetto ai classici applicativi. Infatti, per la realizzazione dell'applicazione ***SyncTrace***, ho fatto molta ricerca con il gruppo di lavoro in merito a nuove tecnologie che potessero essere adeguate al dominio del problema.

La tematica di attualità, contestualizzata con lo scoppio della pandemia, ha giocato un ruolo importante per la decisione degli algoritmi di ***tracing*** e dei ***framework*** da utilizzare; nel mio caso particolare, le scelte relative ai **protocolli di sicurezza** hanno richiesto diversi approfondimenti e molte valutazioni in termini di costo-beneficio, finalizzati a una decisione ponderata per l'esigenza del progetto.

Capitolo 2

Lo stage in Sync Lab

2.1 Lo stage per l'azienda

Lo stage è un momento molto importante nel percorso formativo di uno studente universitario, dal momento che permette di arricchire il proprio bagaglio professionale in modo sostanziale. Le aziende che propongono tirocini, infatti, lo fanno principalmente in collaborazione con l'università dal momento che diventa vantaggioso per i seguenti motivi:

- Da una parte, l'università favorisce il contatto degli studenti con aziende che propongono nuovi sbocchi occupazionali in vista dell'entrata nel mondo del lavoro;
- Dall'altra, l'azienda collabora con gli studenti universitari per portare avanti nuovi progetti, anche di ricerca e sviluppo, o progetti già avviati, offrendo poi nuove opportunità di lavoro con un'apposita selezione.

Calandosi nei panni di Sync Lab, l'azienda ha intenzione di portare avanti l'aspetto di ricerca e sviluppo, sperimentando nuove tecnologie e provando nuove soluzioni che possano poi essere integrate negli applicativi richiesti dall'attuale mercato IT. Personalmente, ritengo che i principali motivi che abbiano portato l'azienda a offrire dei percorsi di stage siano i seguenti:

- **Aprirsi a nuove tematiche:** se il tema che propone l'azienda è interessante nell'ottica di uno studente, sicuramente verrà approfondito in maggior quantità e qualità; tematiche che si rifanno ai contesti di attualità, inoltre, sono ancora più interessanti per l'azienda come forte risposta all'innovazione nel mercato;
- **Nuove tecnologie da impiegare:** l'azienda propone una serie di tecnologie che possono essere utilizzate nella realizzazione di un progetto, ma nella realtà vuole vedere le idee e le opinioni del tirocinante in merito, così da avere una visione nuova e più fresca. In questo modo, sia l'azienda che lo studente possono imparare qualcosa di nuovo, mettendosi in gioco e affrontando insieme nuove tecnologie;
- **Possibilità di assunzione:** per l'azienda è sicuramente importante offrire opportunità lavorative per uno studente che frequenta uno stage, perché permette di selezionare e inserire i giovani nel mondo del lavoro, offrendo loro un assaggio del loro modo di lavorare;

- **Basso impatto economico:** l'università copre quasi totalmente le spese assicurative dello studente e l'azienda non deve investire quasi nulla a livello economico per coprire il tirocinio dello studente.

I vantaggi, pertanto, sono molti, e su questi aspetti Sync Lab ha voluto investire il proprio tempo per mettersi in gioco con molti tirocinanti su un vasto numero di progetti, collaborando con l'Università di Padova, e facendo fronte anche a un periodo molto difficile, che ha richiesto talvolta alcuni compromessi.

2.2 Il contesto di attualità

Dall'inizio del 2020, si è sviluppata una malattia infettiva in Cina a partire dalla città di Wuhan. Il virus della malattia colpisce le vie respiratorie ed è stato identificato come *SARS-CoV-2* o *COVID-19*. La notizia di questo virus si è propagata molto rapidamente in giro per il mondo, tanto che l'OMS (Organizzazione Mondiale della Sanità) ha tenuto d'occhio la situazione per comprenderne bene gli effetti di questo virus e i possibili rischi per la salute.

A partire dalla fine di febbraio 2020, viene annunciato il primo deceduto in Italia da COVID-19¹, un cittadino di Vo' Euganeo in provincia di Padova, il quale, a seguito di un tampone, era stato dichiarato positivo al virus. Successivamente, la stessa città di Vo' è stata messa in quarantena e, con l'aiuto dei medici padovani, sono stati fatti dei tamponi ai cittadini presenti. Questo ha acceso quindi un **primo focolaio** in *Italia*, che ha fatto alzare la tensione a livello nazionale, mettendo in allerta gli ospedali e il governo in vista di possibili misure da adottare per bloccare un possibile contagio.

In seguito, nonostante gli sforzi per contenere il primo focolaio, cominciano ad aprirsi inevitabilmente altri focolai nel nord Italia, più precisamente nelle province in Veneto e in Lombardia, specialmente a Bergamo².



Figura 2.1: P.d.C. Giuseppe Conte durante la pandemia
Fonte: euronews.com

¹Prima vittima da COVID-19 in Italia. URL: <https://www.ilfattoquotidiano.it/2020/02/22/adriano-trevisan-78-anni-di-vo-euganeo-ecco-chi-e-la-prima-vittima-italiana-del-coronavirus-la-paura-nel-paese-isolato/5713686/>.

²Inizio della pandemia e dei focolai in Italia. URL: https://it.wikipedia.org/wiki/Pandemia_di_COVID-19_del_2020_in_Italia#Diffusione_della_pandemia.

La situazione comincia a peggiorare drasticamente nel mondo, e in Italia arriva il *lockdown totale*: tutte le città e tutti gli esercizi commerciali si fermano e vengono chiusi per consentire di ridurre il rischio di contagio. Ogni giorno vengono pubblicati bollettini che illustrano la situazione del contagio e l'andamento della diffusione. Iniziano quindi i controlli attraverso i tamponi su gran parte della popolazione che manifesta i sintomi del virus e viene imposto l'obbligo di rimanere in casa per limitare i rischi di contagio.

In questo periodo, molte aziende, tra cui la stessa **Sync Lab**, decidono di adattarsi e di introdurre lo *smart working* per poter consentire ai propri dipendenti di lavorare in remoto. Anche l'Università di Padova, così come le altre università italiane, mette a disposizione degli strumenti online con cui svolgere le lezioni e continuare con le normali attività insieme agli studenti.

2.3 La risposta dell'azienda



Figura 2.2: Tracciamento dei contatti tramite smartphone

Fonte: huffingtonpost.it

Il modello di vita delle persone è cambiato radicalmente con lo scoppio della pandemia, e questo ha portato alla generazione di nuovi problemi che si sono presentati principalmente con l'evoluzione del virus nel corso del tempo. Uno di questi problemi riguarda la ripresa economica del paese, che ha comportato la riapertura delle città e delle attività commerciali come negozi, ristoranti, bar e alberghi. Buona parte delle persone ha potuto ricominciare a lavorare in presenza, seguendo le dovute precauzioni, rimanendo sempre allerta per eventuali sintomi.

Sync Lab aveva già cominciato a formulare una serie di problemi relativi alla prevenzione del virus, e alcune delle più grandi aziende a livello mondiale, come Apple e Google, avevano già iniziato a studiare il problema, annunciando di avere già qualcosa in sviluppo.

Personalmente, a partire da Marzo avevo già discusso con l'azienda, scambiando alcune opinioni con l'ingegnere Fabio Pallaro riguardo a come sarebbe stato possibile prevenire e tracciare la diffusione del virus, prima che altre persone potessero venire

contagiate. Insieme ad altri colleghi, sono stato coinvolto in alcune riunioni che mi hanno reso partecipe in prima persona della formazione dell’idea in azienda.

Per determinare al meglio il problema, l’azienda ha ragionato su alcune domande chiave molto importanti:

- **Chi sono i principali *stakeholders* dell’applicativo?** Determinare gli attori del sistema ha richiesto all’azienda un gran numero di valutazioni, principalmente focalizzate a studiare anticipatamente i possibili utilizzi futuri anche per operatori sanitari, esercenti e personale aziendale;
- **Che tipo di applicativo realizzare nell’ottica di prevenzione?** Per trovare una risposta a questa domanda, l’azienda ha ragionato approfonditamente su cosa potesse essere facilmente distribuibile e utilizzabile per le persone e per gli *stakeholders*;
- **Quali algoritmi di tracciamento e quali tecnologie utilizzare?** Conoscendo le potenzialità di uno *smartphone*, Sync Lab ha potuto determinare le tecnologie disponibili nella maggioranza dei dispositivi mobili, verificando quindi l’attendibilità delle tecnologie impiegabili e degli algoritmi di tracciamento;
- **Quali dati raccogliere e come gestire le politiche di *privacy*?** Nonostante la necessità di tracciamento, è stato necessario determinare quali fossero i dati effettivamente utili da tracciare e collezionare, studiando poi nel dettaglio la salvaguardia della *privacy* degli utenti;
- **Quali componenti sono necessari per l’applicativo?** Sync Lab ha cercato di identificare fin da subito i componenti richiesti dall’applicativo, così da verificarne il costo in termini di risorse e di tempo per l’effettiva realizzazione.

2.4 Dal problema alla visione del prodotto

Sync Lab ha lavorato a fondo per delineare il problema da studiare e solo successivamente ha determinato gli elementi principali che compongono il prodotto. Partendo dal principio, il problema è stato identificato nell’aspetto di prevenzione del virus, e in particolare nell’uso di una applicazione *mobile* che possa essere facilmente installata dalle persone nei propri *smartphone* per eseguire dei tracciamenti in *background* tramite il *bluetooth*. L’*app* deve permettere di tracciare in forma anonima i contatti (intesi come incontri ravvicinati) con altri *smartphone* che utilizzano la medesima applicazione. Attraverso l’applicazione deve essere possibile preservare un albero dei contatti che non dovrà essere condiviso con il server centrale, a meno di un’autorizzazione da parte dell’utente nel caso in cui quest’ultimo venisse dichiarato infetto. L’accertamento clinico di un infetto sarà a carico degli operatori sanitari, i quali avranno a disposizione una piattaforma web in cui sarà possibile accedere, monitorare i pazienti infetti e richiedere i dati di contatto tramite un processo di autorizzazione. L’applicazione, inoltre, non dovrà richiedere per nessuna ragione il nominativo o altri dati sensibili del paziente, così da mantenere l’anonimato fin dal principio. Questi dati, al più, potranno essere inseriti solamente dai medici per poter monitorare lo stato della malattia.

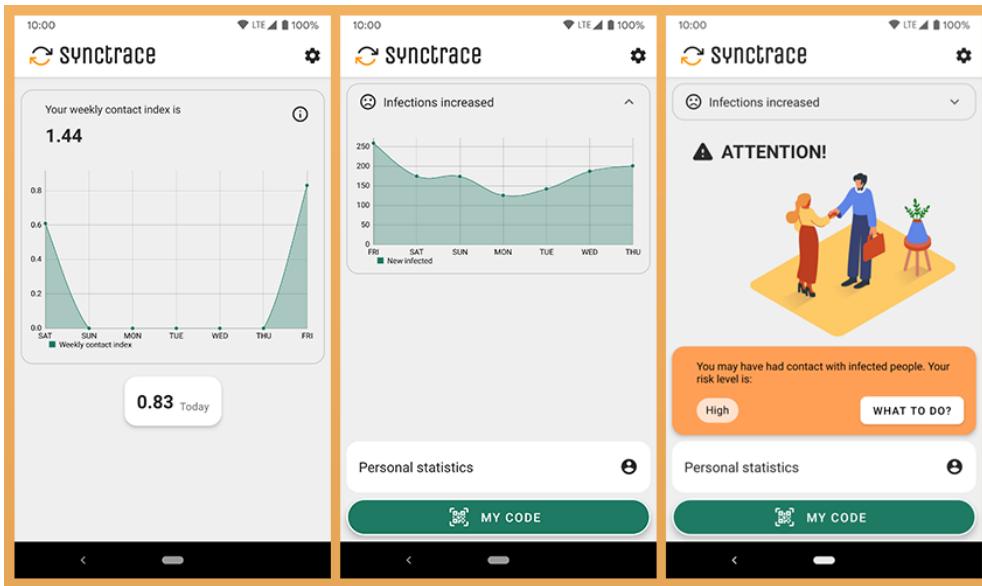


Figura 2.3: Dati sul contact tracing nell'applicazione SyncTrace

Nel momento in cui un utente scoprissse di essere affetto dal virus, a seguito di un apposito controllo mirato, la lista contatti che fornirà agli operatori sanitari previa autorizzazione permetterà di mettere in allerta attraverso una notifica i contatti che sono stati raccolti nel corso delle ultime settimane. In questo modo, solamente le persone a rischio potranno richiedere un tampone con maggiore urgenza, così da verificare il loro stato di salute.

L'applicazione, inoltre, dovrà tenere traccia a livello statistico dei seguenti dati, come riportato nella figura 2.3:

- **Contact Index (CI):** rappresenta l'indice di contatto, ed esprime quanto una persona è venuta a contatto con un'altra in termini di potenza del segnale *Bluetooth Low Energy (BLE)* e durata del contatto. Assume valori da 0 fino a un valore massimo predefinito;
- **Risk Index (RI):** rappresenta l'indice di rischio e fornisce una stima su quanto è probabile che una persona sia stata contagiosa in base al numero di contatti tracciati. L'indice assume valori da 0 fino a un valore massimo predefinito;
- **Risk Level (RL):** identifica il livello di rischio in modo da esprimere in termini più semplici il significato numerico del *Risk Index* per un utente. Le fasce di rischio sono le seguenti: molto basso, basso, medio, alto, molto alto;

Questi dati verranno configurati dal *backend* dell'applicativo per essere ricalcolati in modo giornaliero autonomamente, senza necessità di contattare il server centrale, garantendo così un buon livello di *privacy* per gli utenti.

Sync Lab ha già pensato alle **funzionalità future** che si potranno integrare nel prodotto in vista delle misure di riapertura adottate dal governo³. L'applicazione potrà essere utile per permettere agli esercenti di far accedere i clienti nei propri negozi

³Le misure del governo italiano per l'emergenza COVID-19. URL: <http://www.governo.it/it/coronavirus-misure-del-governo>.

in piena sicurezza. In particolare, si è già pensato di integrare una funzionalità nel portale web per far verificare a un esercente che il cliente non sia stato segnalato come infetto, gestendo così l'accesso nei luoghi chiusi. Oltre a questo, si è pensato di integrare nell'algoritmo di tracciamento una parte con la **blockchain** per salvare i contatti avvenuti con le persone, garantendo così maggiore affidabilità nella parte di tracciamento.

Sync Lab, dunque, ha identificato in modo accurato i principali componenti che dovranno essere realizzati per mettere in funzione l'applicazione:

- **Mobile application:** l'applicazione mobile sarà la parte attiva che raccoglierà i dati degli utenti, rielaborandoli in base ai contatti avvenuti, e fornirà notifiche e *alert* nel caso in cui l'indice di rischio si alzi o nel caso in cui un utente sia entrato in contatto con una persona segnalata come infetta;
- **Web application:** attraverso un portale web, i medici potranno accedere in modo autenticato e sicuro per gestire i dati sensibili degli utenti, monitorando la lista delle persone segnalate come infette e visionando l'albero dei contatti. In futuro, inoltre, gli esercenti delle attività commerciali potranno visionare e verificare se un loro cliente è stato effettivamente infetto, con l'inserimento di un codice casuale fornитогli dal cliente stesso tramite l'applicazione mobile. Questo componente è cruciale che sia solido sotto tutti i punti di vista per poter garantire un livello di sicurezza sufficientemente avanzato;
- **Servizio API REST (backend):** il servizio di *backend* svolgerà le funzioni di controllo delle richieste per accesso alle risorse da parte dell'*app mobile* e del portale web. Questo permetterà di collezionare i dati dell'applicazione mobile, previa autorizzazione dell'utente, e gestire tutte le richieste verso i **database**. Analogamente, tutte le comunicazioni in entrata dovranno essere convalidate in ottica di sicurezza, garantendo accesso alle risorse protette con opportuni accorgimenti che dovranno essere determinati a livello progettuale.

L'azienda quindi ha spinto molto sulla ricerca e innovazione, portando nuove tecnologie come la *blockchain*, il *BLE* per il tracciamento, e l'interazione sincrona con *app mobile* e portale web, preparandosi così a combattere la diffusione del virus.

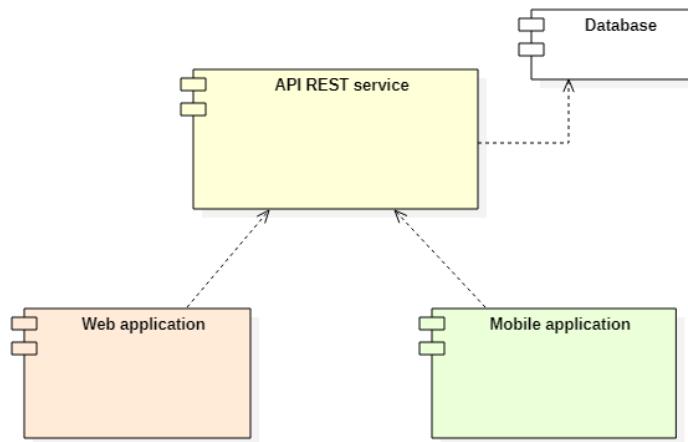


Figura 2.4: Diagramma dei componenti per l'applicazione SyncTrace

2.5 Lo scopo dello stage

Una volta determinati i componenti principali richiesti dal progetto, Sync Lab ha predisposto diversi ambiti formativi in cui coinvolgere più tirocinanti per studiare, progettare e implementare le diverse parti dell'applicativo. Ciascuna componente è stata affidata a più persone, e per ciascuno è stato opportuno studiare un'apposita strategia che permettesse di sviluppare in concorrenza le varie componenti per giungere a un prodotto funzionante, non appena possibile.

Nel mio caso particolare, confrontandomi anche con il tutor aziendale, l'ingegnere Fabio Pallaro, abbiamo determinato insieme una serie di argomenti di studio nel contesto di questo progetto, che potessero essere determinanti in termini di ricerca, sviluppo e innovazione nell'ambito web. Tra i vari aspetti di studio, l'azienda ha richiesto un'analisi dei **protocolli di sicurezza** lato *frontend* nella piattaforma *web* che sarebbe stata sviluppata in concorrenza con l'*app mobile*. Su questo aspetto, l'azienda ha voluto puntare molto nello studio di tecnologie di **autenticazione** e **autorizzazione** che riguardano una *Single Page Application*, sviluppata con il *framework Angular*. Un esempio di queste tecnologie è il protocollo *OAuth 2.0*, ampiamente utilizzato anche da colossi del web come *Google* e *Twitter*, che si compone di un processo di autorizzazione molto avanzato e sicuro, basato su un gran numero di soluzioni architetturali, adattabili in base alle esigenze.

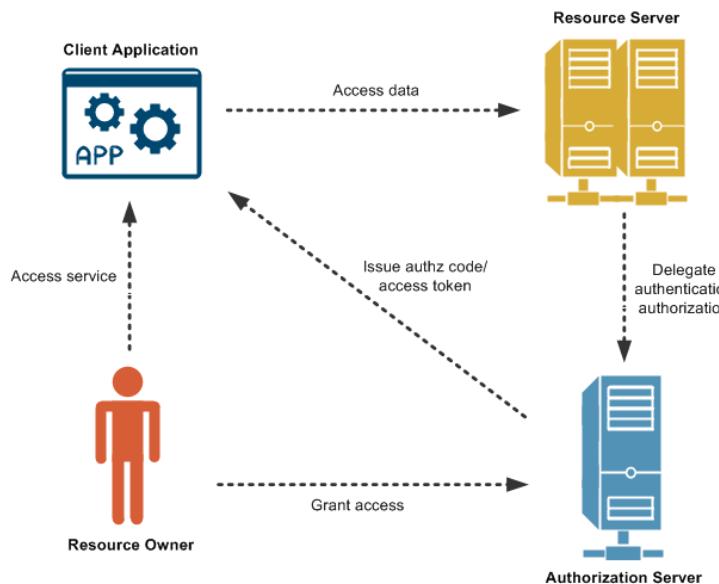


Figura 2.5: Funzionamento ad alto livello di OAuth 2.0

Fonte: docs.oracle.com

A partire da questo studio, ho concordato con l'azienda di voler realizzare un *proof of concept* che potesse mostrare il funzionamento dei protocolli di sicurezza a livello puramente tecnico, determinando quindi il grado di efficacia e di effettiva sicurezza per ciascuno di essi. Sfruttando poi lo stesso *proof of concept* realizzato in fase di progettazione, l'azienda ha richiesto l'implementazione di una **libreria** con il *framework Angular*, così da integrare una soluzione nell'applicativo finale che potesse gestire a

pieno le sessioni di un utente e le richieste verso il *backend*, mediante il protocollo *HTTP*. Da parte mia, pertanto, ho messo sotto esame tutti gli aspetti che riguardavano:

- gli accessi all'interno dell'applicazione web;
- il mantenimento dei *token* per l'accesso alle sezioni protette dell'applicazione web;
- la gestione delle richieste asincrone mediante il protocollo HTTP verso risorse protette nel *backend* (servizio in *API REST*);
- il riconoscimento e il funzionamento dei *Json Web Token*;
- la gestione del *logout* dall'applicativo.

In definitiva, quindi, insieme all'azienda ho concordato una serie di argomenti che potessero aumentare il mio bagaglio di conoscenze e, in particolare, le mie competenze in vista dell'apprendimento di nuovi linguaggi e di nuove tecnologie, specialmente nell'ambito della sicurezza.

2.6 Motivazione della scelta

Ogni anno l'Università di Padova organizza **StageIt**, un evento di incontro per le principali aziende del territorio e per gli studenti. Questo evento favorisce l'ingresso nel mondo del lavoro con offerte di stage che si accompagnano perfettamente con il percorso di studi della Laurea triennale in Informatica. L'evento viene organizzato con l'aiuto del professor Tullio Vardanega in collaborazione con l'associazione degli imprenditori *AssIndustria VenetoCentro*.

StageIT è stato fondamentale per me per poter conoscere molte aziende, così da poter valutare più percorsi di tirocinio e mettermi in gioco nelle tematiche che più mi interessavano. Quest'anno, tuttavia, l'evento è stato molto condizionato dallo scoppio della pandemia, a tal punto da essere organizzato completamente online con opportune videoconferenze e video di presentazione. In questo modo, è stato possibile seguire più aziende e svolgere anche più colloqui, garantendo così una migliore utilizzazione del tempo a disposizione.

Nel mio caso, la scelta di stage è stata direttamente condizionata dagli eventi avvenuti nel corso dell'anno. Con lo scoppio della pandemia, mi sono direttamente interessato alla tematica di **prevenzione**, realizzando anche un sito web per visionare la varianza e i dati ufficiali giornalieri della Protezione Civile riguardo l'evoluzione del virus⁴, e questo mi ha portato a ricercare un percorso che fosse il più attuale possibile. In aggiunta, avevo l'intenzione di approfondire un argomento che potesse riguardare lo **sviluppo web**, specialmente l'integrazione e utilizzo di nuove tecnologie e soluzioni che potessero rispecchiare le principali richieste del mercato IT, in modo da essere preparato anche in vista di un futuro impiego lavorativo in questo ambito. Con il progetto di Ingegneria del Software, infatti, avevo già iniziato a provare nuove tecnologie web, come ad esempio *Laravel* (*framework web* in *PHP* per lo sviluppo di siti web e servizi *REST*) e *NodeJS* (*runtime* in *Javascript* per la realizzazione di servizi web), e pertanto avrei voluto studiare più nel dettaglio un argomento affine, che non riguardasse solamente la creazione di maschere ("view") di un sito, ma che coinvolgesse un'attività di progettazione di servizi utili alla *business-logic* di un sito web.

⁴ Covid-19 - Italy. URL: <https://covid19.debug.ovh/>.

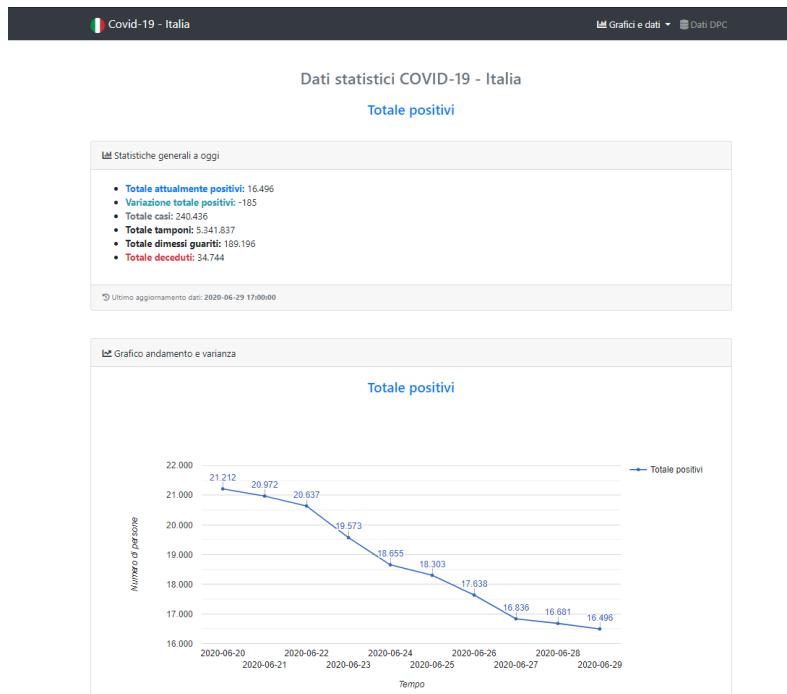


Figura 2.6: Pagina del sito Covid19-Italy per la variazione giornaliera dei dati

Per quanto riguarda il *web security*, lo studio e l’approfondimento dei protocolli di sicurezza sarebbe stato un argomento nuovo che mi sarebbe piaciuto affrontare, posto che, anche nell’ottica di *CyberSecurity*, le proposte lavorative sono potenzialmente molte, e integrare queste nozioni diventa molto vantaggioso. Inoltre, in questo ambito avevo già seguito un corso di CTF (*Capture The Flag*), tenuto dal gruppo *Spritzers* dell’Università di Padova, che trattava delle tecniche di analisi e attacco di applicativi software e servizi web, sfruttando potenziali falle di sicurezza, con *SQL Injection* e *Reverse Engineering*.

Sync Lab mi ha proposto un percorso di stage che avrebbe conciliato tutti gli argomenti, portando all’attenzione un progetto con un prodotto mai visto prima sul mercato, innovativo su molti aspetti e ampiamente alla mia portata. L’azienda, inoltre, ha mostrato grande professionalità e una concreta sicurezza, visto il gran numero di dipendenti, le risorse a loro disposizione e, soprattutto, i clienti con cui hanno collaborato e tutt’ora collaborano nel settore ICT. La loro disponibilità, inoltre, è stata immediata e pertanto la scelta è stata in linea con i campi di ricerca che avrei voluto affrontare per ampliare il mio bagaglio di conoscenze.

2.7 Vincoli e obiettivi dello stage

Per partire bene con il progetto di stage, ho concordato fin dal principio con il *tutor* aziendale gli argomenti, gli obiettivi, i vincoli e la pianificazione temporale per poter trattare a pieno tutto ciò che potesse riguardare la mia parte di prodotto, determinando in modo chiaro e preciso le modalità di interazione remota attraverso lo *smart working*. Il piano di lavoro è stato poi revisionato e confermato sia dal mio relatore, il professor Tullio Vardanega, che dal mio *tutor* aziendale, l’ingegnere Fabio Pallaro.

2.7.1 Pianificazione temporale

Lo stage si è svolto in una durata pari a 8 settimane, per un totale di **300 ore** effettive di lavoro. Come menzionato in precedenza, ho affrontato il tirocinio principalmente in remoto in modalità *smart working*, prevedendo comunque una parte anche in presenza nella sede aziendale, in vista di un miglioramento delle normative sulla sicurezza in termini di prevenzione del virus. Questo mi ha permesso di lavorare da computer per un totale di 6 ore al giorno nelle prime due settimane e 8 ore al giorno nelle successive. La ripartizione settimanale delle attività è stata la seguente:

SETT.	ORE TOT.	ORE	ATTIVITÀ
I	30	6	Videoconferenze con le persone coinvolte nel progetto per discutere precisamente i requisiti e le tecnologie da studiare e impiegare nell'attività di progettazione.
		8	Configurazione degli strumenti di lavoro e della rete aziendale in remoto per accedere a eventuali servizi interni.
		16	Prima analisi delle tecnologie da implementare.
II	30	24	Studio protocolli di sicurezza (<i>OAuthx</i> in primis).
		6	Ripasso di <i>Javascript</i> .
III	40	8	Studio <i>TypeScript</i> .
		16	Studio <i>AngularCLI</i> , <i>Angular</i> , <i>NodeJS</i> .
		16	Realizzazione di un <i>proof of concept</i> .
IV	40	24	Analisi e progettazione della nuova libreria.
		16	Scrittura dei casi di test.
V	40	30	Implementazione della libreria.
		10	Verifica dell'attività di implementazione.
VI	40	30	Implementazione della libreria.
		10	Verifica dell'attività di implementazione.
VII	40	24	Validazione della libreria.
		16	Collaudo finale della libreria.
VIII	40	40	Test di integrazione con il resto dell'applicativo.
	300		<i>Totale ore di stage</i>

Tabella 2.1: Tabella riassuntiva delle ore settimanali di stage

Per quanto riguarda la ripartizione oraria delle principali attività di tirocinio, le singole attività sono state organizzate in modo da coprire totalmente tutte le fasi di sviluppo del prodotto atteso:

- (**14 ore**) Configurazione degli strumenti e delle tecnologie;
- (**86 ore**) Analisi e approfondimento delle tecnologie e dei linguaggi di programmazione (con realizzazione di un *proof of concept*);
- (**40 ore**) Progettazione del prodotto e dei casi di test;
- (**80 ore**) Implementazione e verifica del prodotto;

- (80 ore) Validazione, collaudo finale e integrazione del prodotto.

In presenza, sono state svolte 8 ore lavorative piene, senza particolari restrizioni, ma con opportuni accorgimenti nell'ambiente di lavoro (uso di mascherina, disinfettanti, controllo di temperatura corporea). Le attività in presenza effettive, tuttavia, sono durate solamente 4 giorni, distribuiti nelle ultime 3 settimane di stage, e sono state utili a illustrare attraverso opportuni *meeting* il lavoro svolto fino ad allora.

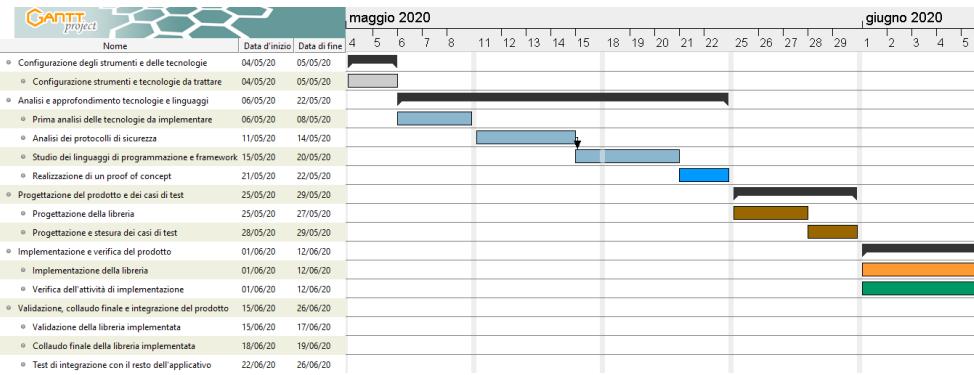


Figura 2.7: Estratto del diagramma di Gantt per la ripartizione delle attività

2.7.2 Vincoli organizzativi

Come riportato precedentemente nella sezione 2.7.1, l'attività di stage si è dovuta svolgere quasi interamente in modalità *smart working*, e ciò ha richiesto una pianificazione dal punto di vista organizzativo che fosse efficace per permettermi di realizzare il prodotto richiesto nei tempi concordati. Ragionando insieme al *tutor* aziendale e insieme al relatore, ho organizzato un piano di lavoro per il tirocinio che potesse soddisfare la necessità di interazione con entrambe le parti, in modo da informare sempre tutti dei miei avanzamenti.

- Per quanto riguarda il **tutor aziendale**, ho organizzato un **registro delle attività**, in cui ho riportato in modo giornaliero le effettive attività svolte nell'arco della giornata lavorativa; queste attività sono state poi controllate e convalidate dal proponente aziendale, garantendo così veridicità del lavoro svolto. Le attività sono state riportate in ordine cronologico e lo stesso registro ha permesso di tenere traccia in modo agile del mio operato.
- Per quanto concerne il **relatore**, ho provveduto a informarlo costantemente dei miei avanzamenti tramite **email** dopo ogni 5 giorni lavorativi effettivi, riassumendo i principali avvenimenti della settimana, illustrando in generale le attività svolte e parlando delle scelte implementative di maggior rilievo che avrei riportato poi nell'elaborato finale.

Riguardo il **registro delle attività**, è stata concordata e redatta la dinamica attraverso cui il processo di registrazione delle attività avrebbe dovuto aver luogo. Nello specifico, si riporta quanto segue:

1. il documento verrà compilato dallo studente ogni giorno (esclusi i *weekend*) dalla data di inizio dello stage e dovrà rispondere ai seguenti quesiti nel dettaglio:

- cosa è stato fatto fino a oggi?
 - cosa deve essere fatto per domani?
 - quali impedimenti rallentano o bloccano i miei progressi?
2. la compilazione del registro avverrà in orario serale non appena verrà conclusa l'attività lavorativa;
 3. il *tutor* aziendale prenderà visione delle attività riportate e convaliderà con un *visto* il documento;
 - a rigore della convalida e della veridicità della *presa visione*, il versionamento supportato dalla suite di *Google Drive* permetterà di controllare e certificare le modifiche apportate al documento da parte dei singoli;
 4. il *tutor* accademico e il *tutor* aziendale potranno prendere visione delle attività riportate in qualunque momento ed esprimere un commento in merito a mancanze o imprecisioni da parte dello studente, se necessario.

Registro delle attività	Tirocinante: Mariano Sciacco	Periodo: 04/05 - 26/06	Tutor aziendale: Ing. Fabio Pallaro
Presa visione ✓	Data	Cosa è stato fatto fino ad oggi?	Cosa deve essere fatto per domani?
<input checked="" type="checkbox"/>	lunedì 25 maggio 2020	<ul style="list-style-type: none"> - Inizio analisi e progettazione della libreria per l'autenticazione e le richieste API in backend - Inizio diagrammi di sequenza, di classe e casi d'uso - Continuazione analisi e progettazione della libreria per l'autenticazione e le richieste API in backend - Continuazione diagrammi di classe, casi d'uso e diagrammi di sequenza - Breve riunione sulle scelte decisionali delle tecnologie a fronte di ulteriori analisi per la webapp - Realizzazione di allineamento sull'avanzamento del lavoro nel progetto - Continuazione analisi e progettazione della libreria (e individuazione dei design pattern) - Continuazione stesura diagrammi di classe e dei package - Inizio idea per i casi di test sulla base dei requisiti individuati 	<ul style="list-style-type: none"> - Continuazione analisi e progettazione della libreria - Continuazione diagrammi di sequenza, di classe e casi d'uso - Nessuno
<input checked="" type="checkbox"/>	martedì 26 maggio 2020	<ul style="list-style-type: none"> - Continuazione analisi e progettazione della libreria - Continuazione diagrammi di sequenza, di classe e casi d'uso - Nessuno 	
<input checked="" type="checkbox"/>	mercoledì 27 maggio 2020	<ul style="list-style-type: none"> - Continuazione analisi e progettazione della libreria - Continuazione diagrammi di sequenza e conclusione diagrammi di classe - Nessuno 	

Figura 2.8: Estratto del registro delle attività giornaliero di stage

Per quanto riguarda il **modello di sviluppo**, Sync Lab fa uso del modello *Agile*, con chiari riferimenti al metodo **Scrum**. Nello stage è stato usato lo stesso modello di sviluppo, e ciò mi ha permesso di apprendere il *modus operandi* dell'azienda e di comprendere sul campo le potenzialità e il funzionamento di uno **Scrum** svolto in modalità *smart working*.

Per quanto concerne le **videoconferenze** e i **canali di comunicazione**, ho deciso insieme al *tutor* aziendale di adottare una comunicazione diretta giornaliera sfruttando strumenti di chat in tempo reale, ampiamente promossi dall'azienda nel periodo della pandemia. In particolare, ho fatto uso di alcune delle tecnologie che avevo già trattato nella sezione 1.4 e che riporto di seguito:

- **Discord**: attraverso questo strumento, ho potuto interfacciarmi direttamente con il *tutor* aziendale sfruttando più canali di comunicazione vocali e testuali, divisi per argomenti. Insieme al gruppo di progetto, ho potuto interfacciarmi anche per confrontarmi nella parte di integrazione della libreria e per collaborare nella progettazione;
- **Google Meet**: per partecipare alle riunioni programmate, ho fatto uso di *Google Meet*, che integra tutte le funzioni di *video conferencing* più importanti e permette di eseguire presentazioni direttamente online. Questo strumento è stato particolarmente utile anche per eseguire uno *scheduling* degli appuntamenti in sincronia con il proprio calendario virtuale;

- **Trello:** attraverso questo strumento ho organizzato preventivamente la lista delle attività e degli *sprint* insieme al *tutor* aziendale. Ogni settimana ho tracciato le attività svolte, inserendo il relativo stato nella bacheca (*Scrum Board*) e monitorando così anche il mio avanzamento in ottica agile;

Le principali riunioni organizzate dal *tutor* aziendale a cui sono stato coinvolto hanno trattato contenuti utili al mio tirocinio sia direttamente, che trasversalmente. Le riunioni sono state di 2 tipi:

- **Riunioni orizzontali** (videoconferenze), per determinare insieme a tutti i membri del progetto i singoli componenti da frammentare e sviluppare per l'applicativo *SyncTrace*, identificando quelli con maggiore priorità e le risorse necessarie per ognuno;
- **Riunioni verticali** (videoconferenze), per determinare singolarmente con un gruppo di lavoro più ristretto l'ambito specifico su cui lavorare, analizzando più nel dettaglio i passi di ogni singolo progetto.

Per quanto concerne la **configurazione**, il proponente aziendale ha messo a disposizione su ***Gitlab*** una *repository* di progetto, in cui è sarebbe stato presente il prodotto per intero. *GitLab*, infatti, è uno strumento molto avanzato che integra *Git* per il versionamento distribuito del prodotto. Nel caso dell'azienda, *Gitlab* è installato nei *server* aziendali (*self-hosted*) e accessibile da remoto con una piattaforma web dedicata. A livello di funzionalità, questo strumento condivide molte delle *features* di *Github*, suo principale concorrente, e permette di integrare una *wiki* nella *repo* di progetto, che è stata utilizzata per mantenere la documentazione accessibile, nonché aggiornata. In aggiunta, *Gitlab* offre anche delle soluzioni ***DevOps*** di *Continuous Integration* e *Continuous Delivery* con dei *server runner* dedicati e configurabili dall'azienda per testare e integrare gradualmente il software su un grande numero di piattaforme diverse.

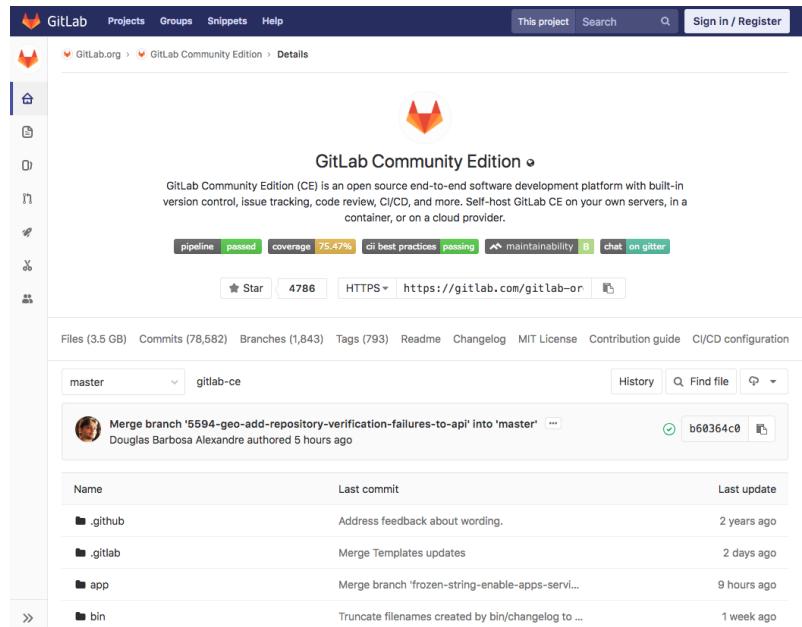


Figura 2.9: Piattaforma web di GitLab

Fonte: wikipedia.org

2.7.3 Obiettivi e prodotti attesi

Insieme al proponente aziendale, ho delineato una serie di obiettivi per quanto concerne i prodotti attesi, i processi coinvolti e la relativa qualità sia di prodotto, che di processo. La nomenclatura decisa per l'identificazione degli obiettivi di stage è formata nel seguente modo:

$$O - [\alpha][\beta]$$

dove:

- α : identifica la tipologia di obiettivo che può assumere i seguenti valori:
 - **O: obbligatorio**, obiettivo vincolante in quanto fortemente richiesto dall'azienda;
 - **D: desiderabile**, obiettivo non vincolante o strettamente necessario, ma dal riconoscibile valore aggiunto;
 - **F: facoltativo**, obiettivo che rappresenta un valore aggiunto non strettamente competitivo;
- β : identifica un numero progressivo relativo alla tipologia di obiettivo.

Gli obiettivi di stage che ho perseguito sono riassunti nella tabella di seguito:

IDENTIFICATIVO	DESCRIZIONE
0-01	Acquisizione delle competenze sulle tematiche di sicurezza frontend e protocolli di sicurezza;
0-02	Acquisizione delle competenze sui linguaggi di programmazione trattati (<i>TypeScript</i> , <i>Javascript</i>) e i relativi <i>framework</i> (<i>Angular</i> , <i>NodeJS</i>).
0-03	Capacità di raggiungere gli obiettivi richiesti in autonomia seguendo il crono-programma.
0-04	Portare a termine l'implementazione della libreria richiesta con una percentuale di superamento pari al 80%.
0-D1	Portare a termine l'implementazione della libreria richiesta con una percentuale di superamento pari al 100%.
0-F1	Dare un contributo effettivo ed efficace sullo studio delle soluzioni presenti sul mercato (<i>community</i>) inerenti alla sicurezza in ambito <i>frontend</i> .
0-F2	Integrare soluzioni di automazione del <i>workflow</i> per l'attività di verifica e validazione del prodotto atteso.

Tabella 2.2: Tabella riassuntiva degli obiettivi di stage

Gli obiettivi fanno riferimento ai prodotti attesi da parte dell'azienda e identificano le attività e gli studi da intraprendere per la realizzazione della libreria in ambito di sicurezza. In riferimento allo scopo dello stage (sezione 2.5), l'azienda ha richiesto i seguenti prodotti organizzati in punti:

1. *Primo punto:*
proof of concept, che mostra il funzionamento della libreria attraverso una

pagina web e fa uso delle tecnologie e dei *framework* scelti a seguito dell'attività di analisi;

2. *Secondo punto:*

documento tecnico della libreria realizzata, con specifica delle classi e dei metodi implementati;

3. *Terzo punto:*

documento di casi di test relativi alla libreria implementata e alla loro integrazione con il resto dell'applicativo;

4. *Quarto punto:*

codice sorgente della libreria implementata, con eventuali file di configurazione per l'automazione del *workflow* nell'attività di verifica e validazione.

A partire dai prodotti, ho evidenziato insieme al proponente aziendale i principali obiettivi in termini qualità che dovranno essere perseguiti a livello di processo e a livello di prodotto.

In primis, trattando degli **obiettivi qualitativi di prodotto**, ho ritenuto opportuno procedere per verifica e validazione sia per quanto riguarda il software, che per quanto riguarda la documentazione, costituendo entrambi un prodotto visto in modo univoco. In particolare, ho definito quanto segue:

- **verifica:** processo di controllo attraverso cui garantisco qualità dei processi di fornitura del mio prodotto;
- **validazione:** processo di controllo del prodotto, volto a confermare le aspettative, i requisiti e le funzionalità concordate.

Attraverso questi processi di controllo garantisco che il mio prodotto rispecchi principalmente i seguenti canoni:

- **adeguatezza funzionale:** il mio prodotto deve essere completo, funzionalmente corretto e appropriato al contesto in cui verrà utilizzato;
- **conformità:** il prodotto deve seguire gli standard e le convenzioni rilevanti, nell'ambito del proprio contesto operativo;
- **usabilità:** il prodotto deve essere fruibile e comprensibile da parte dei principali utilizzatori e deve garantire piena operabilità e protezione dagli errori;
- **sicurezza:** il prodotto deve garantire la sicurezza nella manipolazione dei dati, garantendo integrità e controlli nelle funzionalità più vulnerabili;
- **manutenibilità:** il mio prodotto deve essere modulare e riutilizzabile in base alle necessità dell'azienda;
- **affidabilità:** quanto realizzato deve essere usabile per un lungo periodo di tempo e facilmente riparabile in caso di guasti e ;

Oltre a questo, nel mio *way of working* ho imposto un insieme di obiettivi da conseguire in termini di qualità nell'attività di realizzazione dei test di unità e di integrazione, definendo le soglie minime di percentuale di **Code Coverage**.

I tipi di *code coverage* presi in esame con le relative soglie sono i seguenti:

- ***statements coverage***: misura il numero di *statement* coperti con i test;
- ***branches coverage***: misura il numero di rami condizionali coperti con i test;
- ***functions coverage***: misura il numero di funzioni coperte con i test;
- ***lines coverage***: misura il numero di linee di codice coperte con i test.

TIPO DI COVERAGE	SOGLIA MINIMA	SOGLIA OTTIMALE
<i>Statements coverage</i>	$\geq 75.0\%$	$\geq 90.0\%$
<i>Branches coverage</i>	$\geq 75.0\%$	$\geq 90.0\%$
<i>Functions coverage</i>	$\geq 75.0\%$	$\geq 90.0\%$
<i>Lines coverage</i>	$\geq 75.0\%$	$\geq 90.0\%$

Tabella 2.3: Tabella riassuntiva delle soglie di Code Coverage del prodotto

Per quanto concerne la **qualità di processo**, invece, mi sono impegnato a perseguire nel corso dello sviluppo i principi di efficacia ed efficienza, così da costituire sempre qualcosa di valido e funzionante ad ogni mio incremento. In particolare:

- **efficacia**: il prodotto deve soddisfare tutte le richieste dell'azienda e deve essere convalidato con quanto pianificato;
- **efficienza**: i processi che mi portano alla realizzazione del prodotto devono convergere con costi ridotti in termini di risorse a pari qualità di prodotto, rientrando quindi nel tempo pianificato a mia disposizione.

Una volta fissati tutti i vincoli e obiettivi di stage, ho proceduto con l'avvio effettivo del progetto di stage.

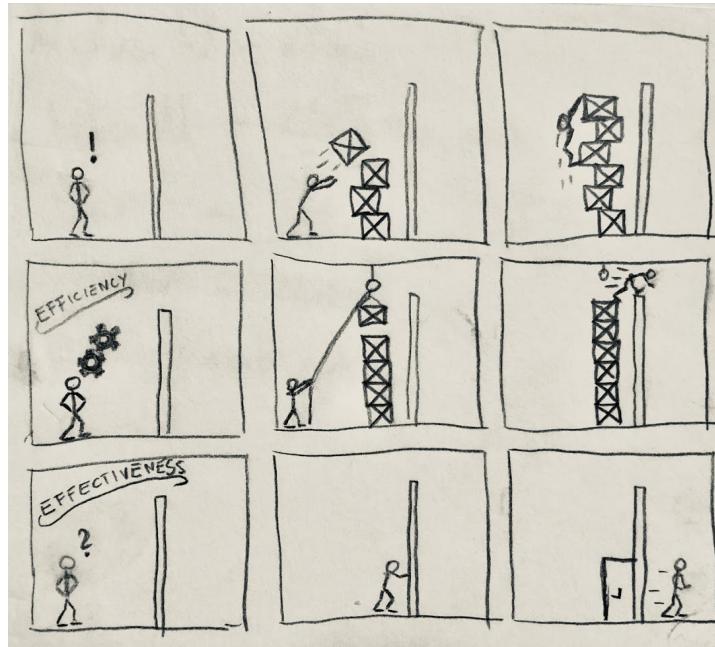


Figura 2.10: Efficienza ed efficacia in una vignetta

Fonte: mbwhatsnew.com

Capitolo 3

Il progetto di stage

3.1 Organizzazione e configurazione

Ho articolato la prima fase del progetto con un processo di organizzazione delle risorse e configurazione degli strumenti di sviluppo, per predisporvi così all'apprendimento e allo sviluppo del prodotto *software*.

Partendo dagli **strumenti**, nel primo periodo ho installato e configurato una serie di applicativi che mi sarebbero tornati utili nel corso dello sviluppo e di cui parte di questi sono già stati utilizzati dall'azienda.

Strumenti di programmazione Nel corso dello sviluppo, ho utilizzato *Visual Studio Code* come editor per lo sviluppo del codice sorgente. Per interfacciarmi con il *framework Angular*, ho eseguito direttamente da linea di comando *AngularCLI*, che mi ha permesso di gestire le singole componenti di progetto, quali classi, interfacce e servizi, in modo automatico. Per la gestione delle dipendenze di progetto, ho utilizzato *Node Package Manager* che mi è tornato utile per aggiungere librerie esterne ed eseguire script automatici per la costruzione e il rilascio del *software*.

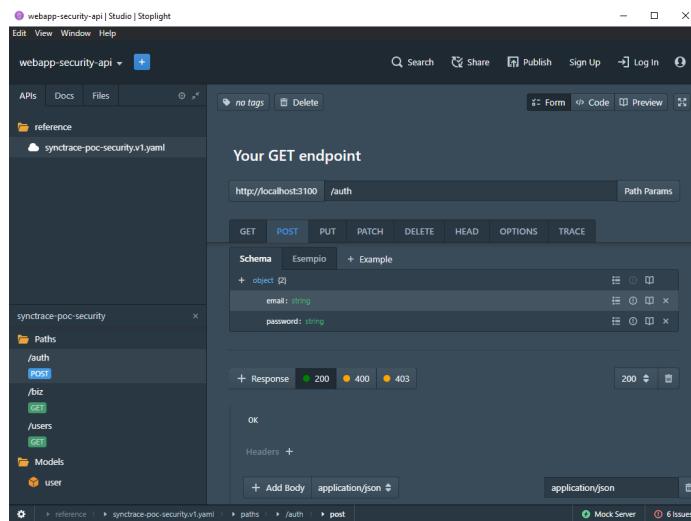


Figura 3.1: Software Stoplight Studio per la definizione delle API

Strumenti di verifica e integrazione Per la verifica delle componenti di progetto, ho utilizzato *Jasmine*, *test suite* nativa di *Angular*, e per eseguire integrazione del *software* ho impiegato il programma *Stoplight Studio* insieme a *Prism*, che mi ha aiutato nella creazione di *server mock* per un servizio *API REST* con dati generati dinamicamente a partire da un documento di specifica *OpenAPI*. Ho sfruttato *TSLint*, infine, per il controllo formale del linguaggio *TypeScript* nel corso dello sviluppo.

Strumenti di versionamento Per il versionamento del software e dei documenti, ho utilizzato *GitLab* sia tramite piattaforma web, accedendo principalmente con *Google Chrome*, che in locale con *GitKraken*, grazie al quale ho potuto gestire sotto il motore *Git* il cambiamento dei file di progetto, tenendo traccia di ogni modifica e lavorando su più *branch* di sviluppo.

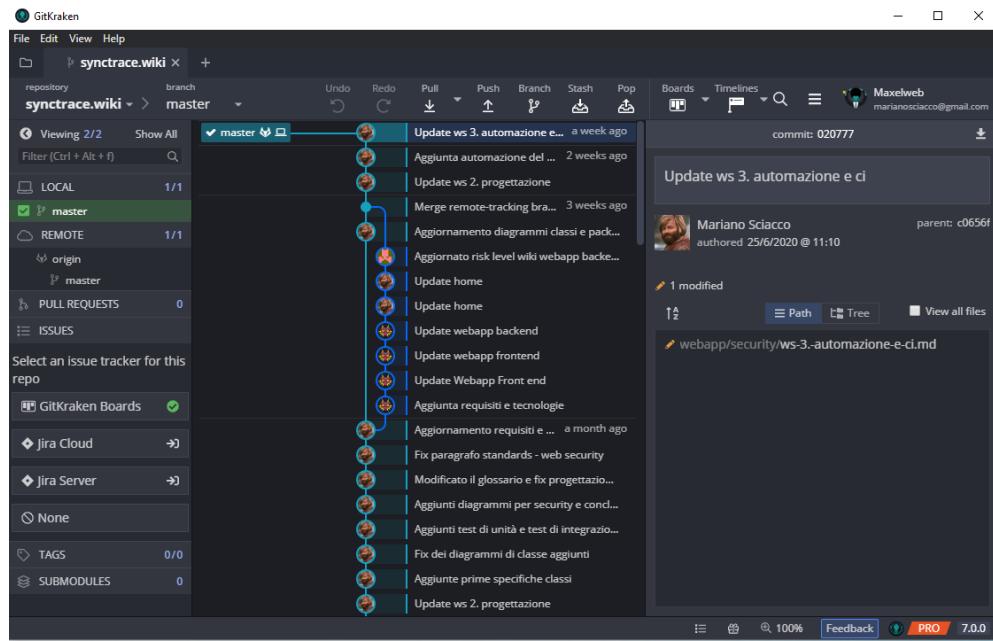


Figura 3.2: Software GitKraken per le operazioni sulla repository di progetto

Strumenti per la documentazione Per la realizzazione di diagrammi *UML* (*Unified Modeling Language*) ho utilizzato *StarUML*, i cui file sono stati salvati e versionati nella *repository* di progetto. Su *GitLab* ho provveduto a realizzare una *wiki* di progetto con il linguaggio *markdown*, la cui accessibilità a questi documenti è stata agevolata a tutti i collaboratori sia per la lettura *online* (tramite *browser*), che per la lettura *offline* (tramite *editor* di testo).

Nel corso della configurazione, ho potuto notare come il *framework Angular* fornisse un ampio parco di programmi nativamente integrati che si orchestrano perfettamente in tutte le loro funzionalità. Oltre a ciò, lo stesso *AngularCLI* prepara allo sviluppatore un *boilerplate* sostanzioso da cui poter partire per lo sviluppo della propria applicazione o per l'estensione di un componente, e ciò ha facilitato di gran lunga sia la parte di formazione, che la parte di implementazione della mia libreria.

Confrontandomi anche con il proponente aziendale, l'ingegnere Fabio Pallaro, l'evoluzione di **Angular** ha introdotto moltissime funzionalità che si sono basate sull'uso di *best practice* fondate sui *design pattern* attualmente più in voga (es. *Facade*, *Decorator*). La solidità di questo *framework* la si ritrova quindi nella sua costante ricerca e attenzione di migliorarsi per stare al passo coi tempi, fornendo agli sviluppatori un ambiente di lavoro già pronto e facilmente interpretabile anche da chi, come me, lo ha imparato in poco tempo.

Per quanto riguarda le **tecnologie** da analizzare, mi sono focalizzato primariamente sui protocolli di sicurezza d'interesse per l'azienda, concordando insieme al *tutor* aziendale una serie di tematiche da approfondire. I principali argomenti, che nelle sezioni successive verranno esposti nel dettaglio, sono stati i seguenti:

Protocollo OAuth 2.0 Utilizzato nell'ambito dei protocolli web, permette di **autorizzare** un'applicazione ad accedere a una serie di risorse protette, fornite da un servizio di terze parti. Generalmente questo protocollo coinvolge un utente che è proprietario di alcuni dati che possono interessare a un'applicazione, e tramite il suo consenso gli stessi dati possono essere acceduti in lettura e/o scrittura. Ad esempio, un'*app* per la gestione della propria rubrica telefonica potrebbe usare questo protocollo per richiedere a *Google* la lista dei contatti dell'utente, in modo da poterla leggere e importare; l'utente quindi dovrebbe accedere con il proprio account *Google* e autorizzare l'applicazione alla lettura della propria lista contatti.

Protocollo OpenID Connect (OIDC) Costruito come *layer* aggiuntivo di *OAuth 2.0*, *OIDC* viene utilizzato per **autenticare** un utente in un'applicazione, sfruttando un servizio di terze parti. A livello pratico, permette dunque di realizzare un servizio di autenticazione a tutti gli effetti, attraverso cui centralizzare gli accessi, soprattutto in caso di applicazioni multiple con più servizi.

JSON Web Token *JWT* rappresenta un *internet standard* per la creazione e gestione di un *payload* di dati tenuti in formato *JSON*. Il *token* viene utilizzato principalmente per conservare dei dati che sono rilevanti per due o più applicativi, in base al contesto architetturale in cui vengono utilizzati. Ad esempio, un *JWT* può essere utilizzato per preservare la sessione di un utente, rendendo un servizio di autenticazione *stateless*, ossia senza bisogno di salvataggi di stato per un particolare utente.

Cookie e Web Storage I *cookie* e il *web storage* rappresentano i principali luoghi di salvataggio delle informazioni lato *client*. Le informazioni, infatti, vengono conservate all'interno dei *browser web*, talvolta per un lungo periodo, e devono essere recuperati dall'applicativo lato *server* per poter interagire correttamente con il resto dell'applicazione. Possono servire, ad esempio, a mantenere attiva la sessione di un utente, e poiché risiedono all'interno del *client* sono le parti più vulnerabili per un applicazione web.

HTTPs e politiche CORS Il protocollo HTTPs (*Hypertext Transfer Protocol over TLS*) è ampiamente utilizzato nelle reti di computer e, in generale, su *internet*, e permette di scambiare informazioni in modo sicuro, estendendo l'attuale protocollo *HTTP*. L'interazione di più risorse tramite il protocollo *HTTP(s)* è regolata da alcune

politiche di condivisione denominate CORS (*Cross-Origin Resource Sharing*), usate per controllare le chiamate verso siti esterni sfruttando gli applicativi lato *client* (generalmente le *Single Page Application* in *Javascript*).

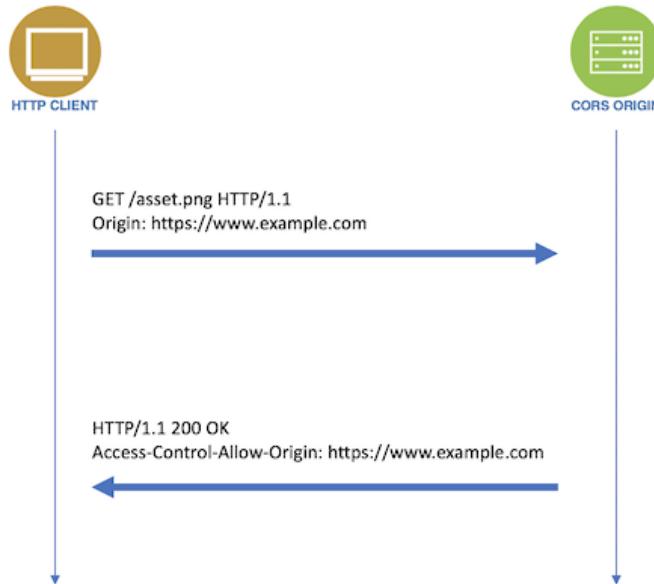


Figura 3.3: Esempio di chiamata HTTP con politica CORS

Fonte: ibm.com

Per quanto riguarda le **riunioni**, ho partecipato a tutte le riunioni fissate con il *tutor* aziendale che mi coinvolgevano sia direttamente, che indirettamente, così da contribuire attivamente allo svolgimento del progetto. Come detto precedentemente nella sezione [2.7.2](#), ho fatto uso di strumenti per le videoconferenze e l'*instant messaging*, come *Discord* e *Google Meet*.

Nel primo periodo, ho definito insieme ad altri colleghi i requisiti di progetto, identificando più a fondo il problema sotto esame. Questo mi ha aiutato a determinare al meglio il campo di studio per la realizzazione della mia libreria e, grazie anche ad alcune riunioni riguardanti le tecnologie, ho potuto apprendere quali strumenti utilizzare e soprattutto come utilizzarli. Nel corso delle settimane successive, mi sono sentito sempre in piena regolarità con il proponente aziendale e, talvolta, anche con il gruppo di lavoro per mostrare gli avanzamenti attraverso le **riunioni verticali**. Riguardo le **riunioni orizzontali**, ho potuto esporre l'avanzamento del mio prodotto agli altri colleghi dell'azienda, ricevendo commenti costruttivi che mi hanno aiutato poi nel corso dello sviluppo. In particolare, le più importanti:

- **Videoconferenza, seconda settimana**, presentazione di nuovi strumenti e tecnologie impiegate dall'azienda, utili allo svolgimento del progetto, tra cui *Stoplight Studio* e *CodeWind* (strumento per generazione del codice sorgente che fa uso delle specifiche *OpenAPI*);
- **Riunione in sede, sesta settimana**, presentazione della libreria implementata fino ad allora, con particolare attenzione alle scelte progettuali e alle *best practice* implementate fino ad allora;

- **Riunione in sede, ottava settimana**, presentazione finale della libreria integrata nell'applicativo *SyncTrace*, e confronto con un altro gruppo di lavoro diverso sulle tecnologie utilizzate.

3.1.1 Automazione del workflow

Per quanto concerne il mio *way of working*, ho configurato l'ambiente di lavoro sulla *repository* di progetto di *GitLab*. Per cominciare, ho predisposto alcune *issue* nell'*issue tracking system* integrato, definendo un *template* da seguire, e impostando le opportune etichette (*label*) per identificare meglio le singole *issue*. Ciascuna *issue*, infatti, poteva rappresentare un argomento di discussione, una nuova funzionalità o un problema (*bug*) da risolvere.

Secondariamente, in vista dell'attività di implementazione e dell'attività di verifica, ho deciso di predisporre, in accordo con il *tutor* aziendale, un *server* che mi permettesse di eseguire in automatico delle attività di **Continuous Integration** della mia libreria, così da essere sempre informato sul suo funzionamento, mediante esecuzione dei test. Per fare ciò, ho approfondito lateralmente la parte di *DevOps*, per potermi dedicare a una corretta configurazione dell'automazione del *workflow*. Questo studio l'avevo comunque previsto negli obiettivi riportati nella sezione 2.7.3, e mi è stato particolarmente utile nell'ultima parte del progetto.

Tuttavia, a causa della pandemia, vorrei precisare di aver messo in funzione il *server* solamente quando ho potuto recarmi in sede aziendale, il che mi ha comunque permesso di coprire una buona parte della verifica e validazione del prodotto.

	superata	#124		P master -> cc29b631 Fix test di integrazione - http	
	fallita	#123		P master -> 00bc56f8 Fix minore libreria SyncSecurity	
	superata	#122		P master -> 4fddddac Merge remote-tracking branch 'origin/mast...'	

Figura 3.4: Esecuzioni della CI nella repository di progetto

Su *Gitlab*, ho potuto configurare dei *runner* che vengono eseguiti in un *server*. Ciascun *runner* rimane in attesa di un *job*, che rappresenta un'attività che il *runner* deve eseguire allo scatenarsi di un evento. Gli eventi possono essere molteplici, ma nel mio caso particolare gli eventi sono stati i seguenti:

- viene eseguito un *push* sulla *repository* di progetto e vengono modificati dei file in una directory specifica;
- richiedo manualmente l'avvio tramite la piattaforma web, a seguito di un fallimento.

Ciascun *runner* viene configurato la prima volta per utilizzare un **esecutore**, che rappresenta il servizio da invocare che è disponibile all'interno del *server*. In questo caso, ho fatto uso di **Docker**, che è uno dei servizi principali per l'avvio del *software* in un ambiente delimitato e controllato (*container*), che può essere facilmente scalabile e facilmente installabile in un gran numero di soluzioni. Tramite *Docker*, ho configurato

un *container* in cui è stata "virtualizzata" la mia applicazione, senza alterare lo stato del sistema operativo ospitante (*host*).

La configurazione della *CI* su *GitLab* avviene attraverso la definizione di un file `.gitlab-ci.yml` inserito nella *root directory* del progetto. Una estratto della configurazione è il seguente:

Codice sorgente 3.1: Esempio di un file di configurazione CI su GitLab

```
# -----
# Configurazione CI - Webapp
#       Gitlab Runner
# -----


# [...]

test_webapp_security:
  image: springmedia/rbbt-fnk-docker-node-chrome-headless:chr78-n12.13.0-y1.19.1
  script:
    - cd webapp/webapp-security
    - npm install
    - nohup npm run mockapi &
    - npm run test-ci-headless
  only:
    changes:
      - webapp/webapp-security/**/*
```

Attraverso questa configurazione, vengono eseguiti i seguenti passaggi:

1. Preparazione del *container Docker* (definito in `image`);
2. Aggiornamento e clonazione delle ultime modifiche della *repository*;
3. Installazione delle dipendenze di *npm* (`npm install`) nella cartella del progetto *Angular*;
4. Avvio di eventuali servizi integrativi in *background* (`nohup npm run mockapi &`) per il test con le *API*;
5. Esecuzione del processo di *build* e *test* con un'istanza di *Google Chrome headless*;
6. Pubblicazione dei risultati dei test e del *Code Coverage*;
7. Conclusione e notifica del responso via *Email*.

L'intero processo di integrazione si svolge sulla macchina messa a disposizione con *Docker*, mentre i *logs*, ossia gli *output* del *runner*, possono essere visualizzati in tempo reale dalla piattaforma web di *GitLab*. Al termine di ogni processo, il *container docker* viene aggiornato e pulito, permettendo così di ricreare dinamicamente l'ambiente di sviluppo.

The screenshot shows a GitLab interface with a pipeline log for a project named 'test_webapp_security'. The log displays a series of test executions (288 to 306) for 'Chrome Headless' on 'Linux x86_64'. Each test is marked as 'SUCCESS' with execution times ranging from 0.63 to 0.868 seconds. A 'TOTAL' row shows 83 successes. Below the tests, a 'Coverage summary' section provides detailed coverage statistics for statements, branches, functions, and lines, all showing high coverage rates (e.g., 90.16%, 76.25%, 98.57%, 89.78%). The pipeline status is marked as 'Job succeeded'.

```

288 Chrome Headless 78.0.3904.97 (Linux x86_64): Executed 80 of 83 SUCCESS (0 secs / 0.63
2 secs)
289 Chrome Headless 78.0.3904.97 (Linux x86_64): Executed 81 of 83 SUCCESS (0 secs / 0.63
5 secs)
290 Chrome Headless 78.0.3904.97 (Linux x86_64): Executed 82 of 83 SUCCESS (0 secs / 0.63
8 secs)
291 Chrome Headless 78.0.3904.97 (Linux x86_64): Executed 83 of 83 SUCCESS (0 secs / 0.64
secs)
292 Chrome Headless 78.0.3904.97 (Linux x86_64): Executed 83 of 83 SUCCESS (0.868 secs /
0.64 secs)
293 TOTAL: 83 SUCCESS
294 TOTAL: 83 SUCCESS
295 TOTAL: 83 SUCCESS
296 ===== Coverage summary =====
297 Statements : 90.16% ( 174/193 )
298 Branches   : 76.25% ( 61/80 )
299 Functions  : 98.57% ( 69/70 )
300 Lines      : 89.78% ( 167/186 )
301 =====
306 Job succeeded
  
```

Figura 3.5: Logs della CI nella repository di progetto

Per l'attività di verifica, ho ritenuto fondamentale l'impiego di questo processo di automazione, che mi ha permesso di convalidare continuamente il funzionamento del mio prodotto a seguito di un nuovo incremento. Ciò ha fatto molto piacere al *tutor* aziendale, dal momento che a partire da questa configurazione ho reso possibile configurare anche altri ambienti inerenti allo stesso progetto, fornendo così un riscontro più istantaneo sul funzionamento dei vari componenti.

3.2 Analisi dei protocolli di sicurezza

Nel corso delle prime settimane, ho concordato con il *tutor* aziendale gli argomenti da approfondire per quanto concerne le tecnologie e i protocolli di sicurezza, utili al contesto applicativo del progetto *SyncTrace*. Nello specifico, ho approfondito nel dettaglio tutta la parte di sicurezza dei servizi web, concentrandomi sulla sicurezza a livello di applicazione web con il *framework Angular*. Per questo motivo, ho ricercato autonomamente gli argomenti di maggiore interesse, documentandomi con opportuni **pro e contro** e preparandomi a portare un *proof of concept* su cui avrei basato le mie scelte progettuali per la libreria.

3.2.1 Protocollo OAuth 2.0

Il protocollo *OAuth 2.0* è uno standard industriale utilizzato per l'accesso alle risorse protette di un servizio terzo, attraverso l'impiego di un processo di **autorizzazione**. Questo processo viene definito per intero dal protocollo e si può basare su diverse tipologie di *flow*, che si adattano alle esigenze architetturali del *software* in cui deve essere implementato. *OAuth 2.0* è largamente utilizzato nel panorama del mercato *IT*, tanto da essere utilizzato anche da alcuni giganti del web come *Google*, *Twitter* e *Facebook*. Il protocollo può essere implementato in diverse tipi di applicazioni, tra cui siti web, ma anche in applicazioni desktop e *app mobile*.

La sicurezza di questo protocollo risiede nell'obiettivo primario di rendere il più trasparente possibile all'utente i permessi che può concedere a una data applicazione, nascondendo la complessità delle chiamate che vengono effettuate lato *client* e lato

server. Questo protocollo, quindi, definisce uno standard di comunicazione tra due o più applicazioni per scambiarsi informazioni a partire da un processo di **autorizzazione** (da non confondere con il processo di **autenticazione**, utilizzato nel protocollo *OpenID Connect*).

Per comprendere bene il suo funzionamento, riporto un esempio concreto di utilizzo del protocollo, prima e dopo una sua implementazione in un applicativo.

Are your friends already on Yelp?

Many of your friends may already be here, now you can find out. Just log in and we'll display all your contacts, and you can select which ones to invite! And don't worry, we don't keep your email password or your friends' addresses. We loathe spam, too.

The screenshot shows a web interface for checking contacts. At the top, there's a section titled "Your Email Service" with radio buttons for "msn Hotmail", "YAHOO! MAIL", "AOL Mail", and "Gmail". Below that, there are fields for "Your Email Address" containing "ima.testguy@gmail.com" and "Your Gmail Password" with masked input. A note says "(The password you use to log into your Gmail email)". At the bottom, there are two buttons: "Skip this step" and a yellow "Check Contacts" button.

Figura 3.6: Una funzionalità del sito web Yelp

Fonte: Nate Barbettini, speakerdeck.com

In origine, siti web come *Yelp* permettevano di inviare delle email ai propri contatti presenti sul proprio *account Google*, richiedendo direttamente le credenziali di accesso, come è possibile vedere nella figura 3.6. Il processo di invio email era automatizzato, ma questo ha richiesto agli utenti di fidarsi di *Yelp*, posto che i dati inseriti erano altamente sensibili e davano accesso a tutte le informazioni del proprio *account Google*.

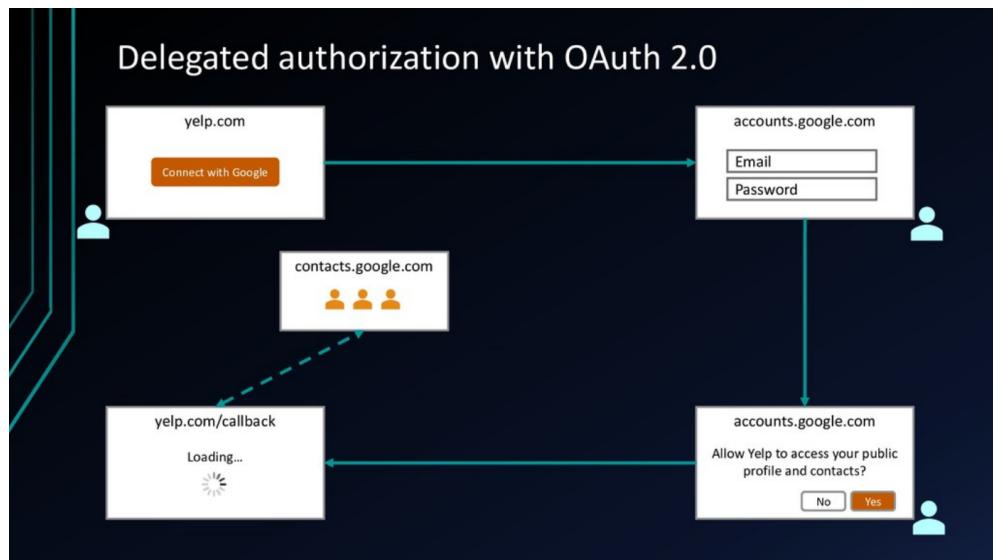


Figura 3.7: Autorizzazione OAuth 2.0 su Yelp

Fonte: Nate Barbettini, speakerdeck.com

Con l'introduzione di *OAuth 2.0*, invece, gli sviluppatori hanno voluto **evitare** questa pratica in favore di un processo di autorizzazione attraverso cui è l'utente che decide nello specifico quali permessi concedere a una determinata applicazione. Questo ha permesso di garantire una maggiore **protezione dei dati**, dando consapevolezza e trasparenza all'utente sul trattamento delle sue informazioni.

Implementando *OAuth 2.0* la parte di autorizzazione dei contatti su *Yelp* avviene attraverso il seguente processo (rif. figura 3.7):

1. l'utente clicca sul bottone "*Connect with Google*";
2. viene eseguito un *redirect* sul sito di *Google* in cui vengono richieste le credenziali di accesso;
3. una volta eseguito l'accesso, viene richiesto il consenso dell'utente per l'utilizzo di tutti o di parte dei suoi dati (*scopes*) presenti nel proprio account (es. i propri contatti);
4. viene eseguito un *redirect* sul sito *Yelp*, che adesso otterrà accesso ai contatti dell'*account Google* dell'utente con un'opportuna autorizzazione.

L'intero processo si articola in due canali di comunicazione, che svolgono le funzioni attive per l'intero processo di consenso e autorizzazione dell'utente all'uso dei suoi dati. In particolare i due canali sono:

- il **front channel**, ossia il canale del *client* (in questo caso il *browser web* dell'utente) che sfrutta componenti come gli *url redirect* per il passaggio dei dati;
- il **back channel**, ossia il canale del *server*, in cui gli applicativi comunicano tra di loro con delle comunicazioni interne *server-to-server*, molto più sicure rispetto al *front channel*.

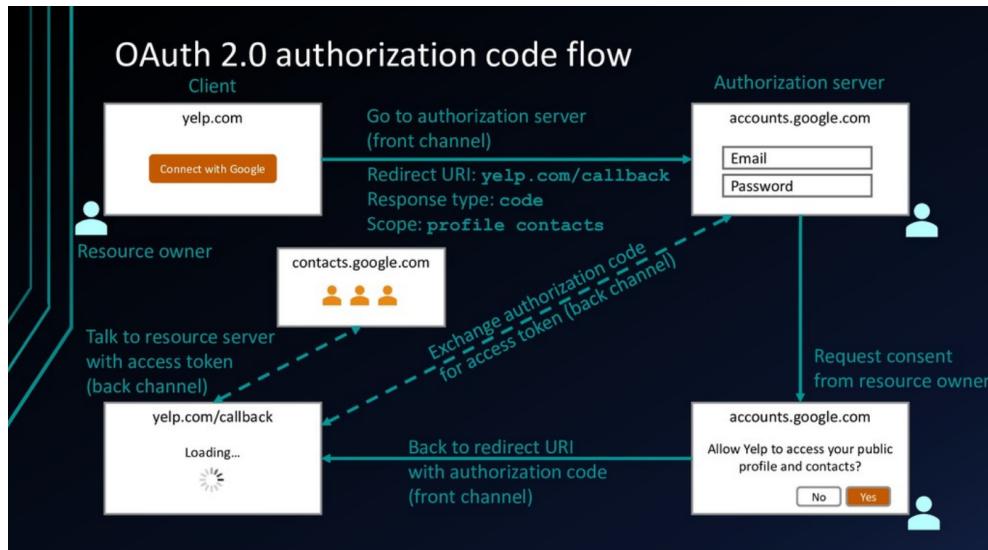


Figura 3.8: OAuth 2.0, Code flow nel dettaglio

Fonte: Nate Barbettini, speakerdeck.com

Scendendo nel dettaglio, a partire dallo studio che ho fatto sul protocollo, ho identificato i componenti ricorrenti che vengono coinvolti nel processo di autorizzazione:

- **Resource Owner**: identifica l'utente proprietario dei dati (nel caso dell'esempio, l'*account Google* dell'utente).
- **Resource Server**: identifica il *server* in cui sono presenti i dati (nel caso dell'esempio, i *server database* di *Google*).
- **Authorization Server**: identifica il *server* a cui si vuole accedere per la lettura e/o scrittura dei dati (nel caso dell'esempio, la pagina `accounts.google.com` in figura 3.8).
- **Client**: identifica il programma che utilizza l'utente per interfacciarsi col protocollo (nel caso dell'esempio, il *browser web*).
- **Authorization Grant**: identifica il consenso dell'utente per l'accesso a un determinato *scope* dei propri dati (nel caso dell'esempio, il "sì" o "no" al consenso).
- **Access Token**: identifica un *token* di autorizzazione, attraverso cui l'applicativo può interfacciarsi autonomamente *server-to-server* per l'accesso ai dati consentiti dall'utente (nel caso dell'esempio, il collegamento *callback* alla pagina `contacts.google.com` in figura 3.8).
- **Scopes**: identifica l'insieme di permessi per la lettura e scrittura di dati presenti nel *resource server* (nel caso dell'esempio, i contatti dell'*account Google*).
- **Redirect URI**: identifica l'*url* di redirezionamento al termine del processo di autorizzazione (nel caso dell'esempio, `yelp.com/callback`).

Precedentemente ho menzionato che ci possono essere più *flow*, ossia più tipi di adattamenti di questo protocollo. Il percorso di autorizzazione *standard*, denominato *authorization code flow*, riassume molto bene il funzionamento generale del protocollo e può essere descritto più formalmente nel seguente modo:

1. Il **client**, nonché **resource owner**, procede nell'**authorization server** con un *redirect URI*, una serie di *scopes* e un *response type*, che identifierà l'oggetto di ritorno (**code**) alla pagina di *redirect*;
2. Nell'*authorization server*, l'utente inserisce le proprie **credenziali** (*front channel*);
3. Se le credenziali sono corrette, il *server* richiede un **authorization grant** agli *scopes* che sono stati richiesti dall'applicazione;
4. Se l'utente acconsente, viene ricaricata la pagina verso il **redirect URI** specificato inizialmente, con in allegato un **authorization code** generato dall'*authorization server*;
5. Una volta eseguito il *redirect*, l'applicazione riceverà in chiaro l'*authorization code*, che verrà scambiato per un **access token** con una comunicazione *server-to-server* (*back channel*);
6. L'applicazione salverà l'*access token* ricevuto e lo utilizzerà per eseguire le richieste agli *scopes* autorizzati dall'utente, contattando direttamente il servizio di riferimento.

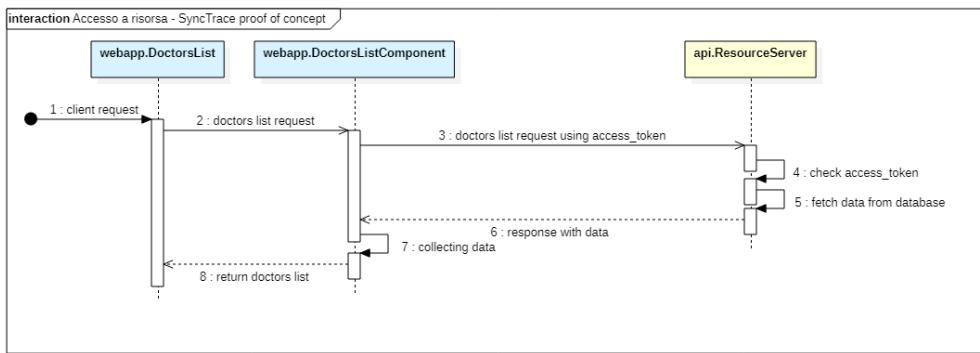


Figura 3.9: Diagramma di sequenza per l'accesso a una risorsa protetta

Nel caso di *SyncTrace*, ho ritenuto che il protocollo *OAuth* avrebbe potuto gestire la parte di autorizzazione per l'accesso alle risorse nel momento in cui vengono eseguite delle chiamate al **resource server** (figura 3.9), dopo aver richiesto le opportune autorizzazioni con il processo illustrato precedentemente (figura 3.8).

Scendendo nel dettaglio, dal momento che la piattaforma web è realizzata in *Angular*, tutta la parte del *back channel* risulta essere mancante, e questo ha portato a un primo problema a livello di sicurezza. A differenza di altri applicativi come *Laravel* o *PHP*, *Angular* affida gran parte dell'accesso ai dati a un servizio *API REST*, eliminando la possibilità di eseguire alcune operazioni di *business-logic* utili come in questo caso.

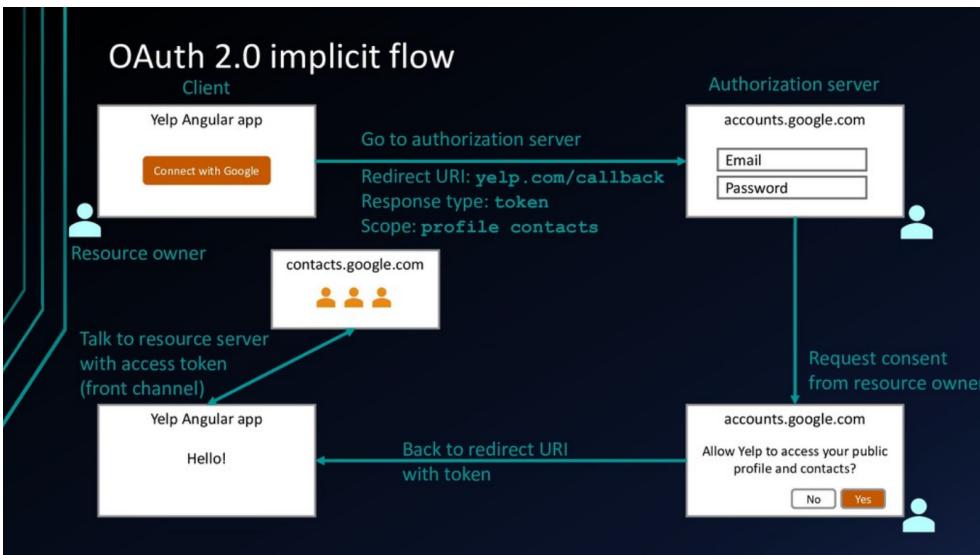


Figura 3.10: OAuth 2.0, dettaglio Implicit flow

Fonte: Nate Barbettini, speakerdeck.com

Fortunatamente, *OAuth 2.0* si adatta anche a questa casistica con l'*implicit flow*, come definito nella specifica RFC-6749¹. Questo percorso di autorizzazione, infatti, rimuove la necessità di avere una comunicazione *server-to-server* e un *authorization*

¹ The OAuth 2.0 Authorization Framework, Implicit Flow, RFC 6749. URL: <https://tools.ietf.org/html/rfc6749#section-1.3.2>.

code, fornendo direttamente all'applicazione web l'*access token*, dopo l'*authorization grant* (vedasi figura 3.10). Tale pratica, tuttavia, riduce anche la **sicurezza**, dal momento che gran parte delle operazioni avvengono nel *front channel*, e ciò porta a maggiore esposizione di possibili errori e falle nell'applicativo.

Dal momento che l'esigenza del progetto mi ha spinto a ricercare anche un servizio che fornisce **autenticazione**, oltre che autorizzazione, ho avuto modo di approfondire le principali estensioni di *OAuth 2.0*, analizzando in particolare il protocollo *OpenID Connect* (OIDC), al fine di comprendere al meglio i casi d'uso e l'effettiva necessità di integrazione di questi protocolli nel contesto architetturale del prodotto finale.

3.2.2 Protocollo OpenID Connect

Il protocollo *OpenID Connect* (OIDC) è un'estensione costruita al di sopra di *OAuth 2.0*, che viene utilizzata per gestire l'autenticazione e l'identità nelle applicazioni web, mobili e *Javascript-based*. Questo protocollo permette di richiedere e ricevere dati inerenti alle sessioni, definendo per intero un processo attraverso cui un utente può autenticarsi in un servizio terzo e può visualizzare le proprie informazioni.

Nel passato, il protocollo *OAuth 2.0* è stato usato impropriamente per l'implementazione di servizi di identità basati esclusivamente su *access token*², e questa è stata la spinta per fare maggiore chiarezza per fornire agli sviluppatori una soluzione che potesse essere il più adeguata possibile alle loro esigenze.

L'introduzione di *OpenID Connect* ha risolto un gran numero di problematiche relativi ai problemi di sicurezza che insorgevano con gli applicativi *OAuth 2.0* usati per l'identità, evitando di complicare comunque l'aspetto implementativo del nuovo protocollo in favore della semplicità e della compatibilità, almeno parziale.

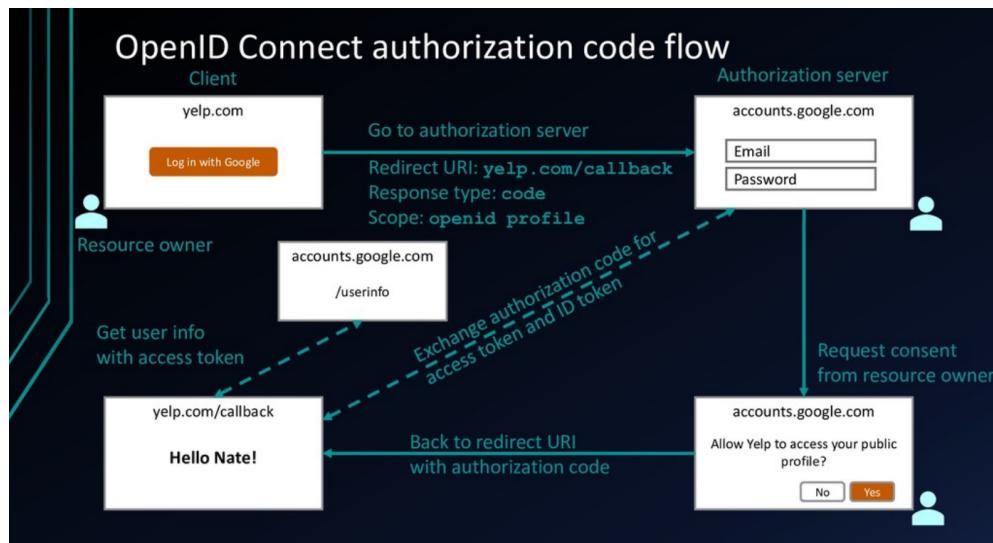


Figura 3.11: OpenID Connect, Code flow

Fonte: Nate Barbettini, speakerdeck.com

Rispetto a *OAuth 2.0*, *OpenID Connect* introduce i seguenti cambiamenti:

²End User Authentication with OAuth 2.0. URL: <https://oauth.net/articles/authentication/>.

- utilizzo di un ***ID token***, in aggiunta all'*access token*, per gestire interamente la sessione dell'utente sfruttando il *JSON Web Token*;
- realizzazione di ***endpoint*** (*User Info*) *ad hoc* per reperire le informazioni di un singolo utente;
- ***scopes standardizzati***, ossia set di permessi che l'utente può consentire, regolati in modo standard.

Come è possibile vedere nella figura 3.11, il protocollo rimane pressoché simile a *OAuth 2.0*, con le principali differenze identificate nello *scope* di richiesta iniziale e nello scambio di un *authorization code* per l'ottenimento di un *ID token* e di un *access token*.

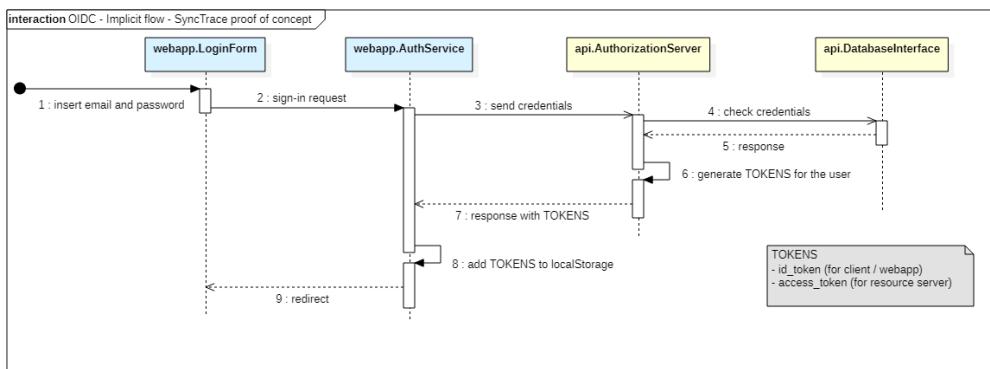


Figura 3.12: Diagramma di sequenza di per l'accesso con il protocollo OpenID Connect

Nel caso del progetto *SyncTrace*, ho valutato una possibile implementazione di questo protocollo in quanto servizio di identità e, analizzando il caso d'uso a fondo, ho determinato che per funzionare sarebbe stato necessario abbandonare il *code flow* in favore dell'*implicit flow* che, come detto precedentemente, risulta essere meno sicuro e molto più oneroso da implementare, specialmente in questo caso. Questo protocollo, tuttavia, lo ho tenuto in considerazione in vista di una sua realizzazione parziale nel *proof of concept*, così da comprendere a pieno il funzionamento a livello tecnico e a livello implementativo, valutandone così il costo-beneficio in termini di sicurezza per la *web app*.

3.2.3 JSON Web Token

Direttamente collegato con il protocollo *OpenID Connect* e *OAuth 2.0*, il *JSON Web Token* è uno standard internet per la trasmissione di dati in formato *JSON* (*Javascript Object Notation*). Questo standard possiede alcune caratteristiche molto importanti che lo rendono particolarmente utile nel contesto web.

Compattezza Il *JWT* è sufficientemente compatto per essere utilizzato negli *url* e nelle intestazioni (*header*) del protocollo *http*, riducendo così lo spazio necessario per l'invio delle informazioni.

Formato dati I dati contenuti nel *payload* e nell'intestazione del *token* sono organizzati in formato *JSON*, ampiamente utilizzato nel contesto web e facilmente adattabile in moltissimi contesti applicativi.

Autonomia Il *token* è formato da più parti che identificano la sua tipologia, il contenuto ("*payload*") e la firma che lo convalida. Per questo motivo un *JWT* può contenere anche informazioni inerenti alla sessione dell'utente senza necessità di richiesta da parte dell'applicativo al servizio *backend* per la convalida.

Sicurezza Il *JWT* può essere firmato con un algoritmo che viene definito nell'intestazione e pertanto le informazioni riportate al suo interno non possono subire alterazioni; il *token* inoltre può essere verificato e controllato dall'emittente per garantire la sua validità.

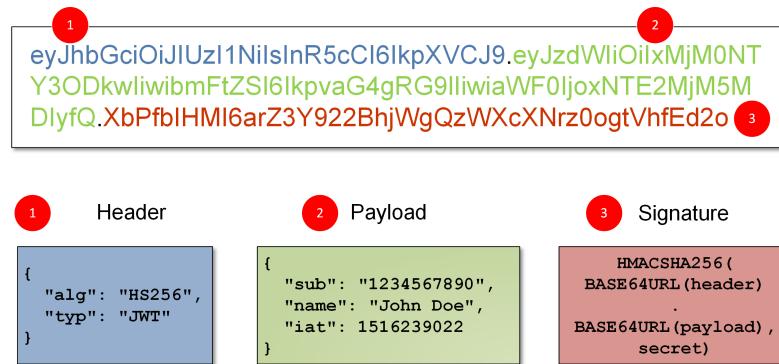


Figura 3.13: Struttura di un JWT
Fonte: securitum.com

Un *JWT*, per definizione del documento RFC 7519³, è formato da 3 elementi principali che vengono separati da un punto. Ciascuno di questi identifica in modo ordinato:

- un'**intestazione** (*JOSE Header*, ossia *Javascript Object Signing and Encryption Header*), in cui sono presenti il tipo di *token* (*typ*) e l'algoritmo per la generazione della firma (*alg*), oltre ad altri campi opzionali;
- un **corpo** (*payload*), dove è presente il dato per intero in formato *JSON* che può essere letto sia lato *server*, che lato *client*;
- una **firma** (*signature*), che viene generata a partire dall'intestazione e dal corpo del *token*, integrando talvolta un *secret* o una chiave pubblica per la verifica dell'emittente.

Generalmente, nel corpo del *token* vengono inserite anche delle asserzioni (*claims*), pubbliche o private, che possono contenere informazioni importanti relative al *token* ai fini di sicurezza. Di seguito riporto alcune delle asserzioni pubbliche più rilevanti:

³JSON Web Token, RFC 7519. URL: <https://tools.ietf.org/html/rfc7519>.

CLAIM	SIGNIFICATO	DESCRIZIONE
<code>iss</code>	<i>issuer</i>	emittente del <i>token</i>
<code>sub</code>	<i>subject</i>	soggetto univoco del <i>token</i>
<code>exp</code>	<i>expiration time</i>	tempo di scadenza del <i>token</i>
<code>iat</code>	<i>issued at</i>	data di emissione del <i>token</i>
<code>nbf</code>	<i>not before</i>	tempo prima del quale il <i>token</i> non va utilizzato
<code>jti</code>	<i>JWT ID</i>	identificatore univoco del <i>token</i>
<code>aud</code>	<i>audience</i>	pubblico a cui è rivolto il <i>token</i> (ossia i casi per cui il <i>token</i> deve essere utilizzato)

Tabella 3.1: Tabella riassuntiva delle asserzioni principali di JWT

Per la generazione di un *JWT*, lo standard utilizza l'*encoding* in **base64**, ma la generazione della firma del *token* fa uso di algoritmi di *hashing* (o simili, come le curve ellittiche), definiti nell'intestazione, come ad esempio *HS256* (ossia *HMAC in SHA-256*). Il *token*, in alternativa, può essere crittato per preservarne le informazioni sensibili con opportuni algoritmi riportati sempre nell'intestazione. A partire dal *JOSE Header* del *token*, il *JWT* può essere identificato in 2 tipologie:

- **Json Web Signature** (*JWS*) che rappresenta un *JWT* con una firma generata *ad hoc* in base al contenuto;
- **Json Web Encryption** (*JWE*) che identifica un *JWT* crittato con un algoritmo riportato nell'intestazione.

Entrambi i tipi di *token* possono essere impiegati in ambiti applicativi differenti, in base allo scopo per cui sono stati pensati. Nel caso d'uso di una sessione, ad esempio, può essere comodo utilizzare un **JWS**, dal momento che questo permette al *client* (l'applicativo web) di poter leggere direttamente delle informazioni contenute nel *payload* senza dover eseguire alcuna chiamata aggiuntiva alle *API*. Per contro, se dovessi trasferire dei dati sensibili tra due applicazioni, come codici di accesso per un *Two Factor Authentication*, un **JWE** potrebbe essere una soluzione appropriata, posto di essere a conoscenza dell'algoritmo di criptazione e dei requisiti di decriptazione.

Nell'ambito del progetto, ho studiato approfonditamente il *JSON Web Token* per poterlo integrare nel mio prodotto finale, essendo un invariante molto ricorrente per l'aspetto di sicurezza. A differenza di altri concorrenti per la trasmissione dati come *SAML* (*Security Assertion Markup Language*), che utilizza *XML* (*eXtensible Markup Language*) per la struttura dati, *JWT* si concentra sulla semplicità e sulla compattezza, portando come punti di forza anche il formato dei dati, *JSON*, e la compatibilità con più applicativi e *framework*.

3.2.4 Principali vulnerabilità di OAuth 2.0

A livello di sicurezza, *OAuth 2.0* è parecchio complesso, e trova le sue fal当地 maggiore nei *flow* che mette a disposizione. L'*authorization code flow* risulta essere tra i *flow* più solidi, dal momento che:

- il passaggio di autorizzazione dell'utente nel **front channel**, che è la parte più vulnerabile, viene assicurata con l'*authorization code*, emesso dall'*authorization server*;

- il passaggio *server-to-server* nel ***back channel***, invece, è controllato dallo scambio di un *authorization code* in favore di un *access token*.

Per quanto riguarda l'*implicit flow*, i maggiori problemi di sicurezza si ritrovano nella richiesta del *token*, che viene inviato in chiaro nel *browser* dell'utente. Alcuni dei principali problemi sono esposti di seguito.

Cronologia in Cloud Il *token* viene inviato nell'*url* di ritorno e i *browser web* salvano la cronologia dell'utente talvolta in *cloud*; questo può essere dannoso dal momento che tramite un attacco sarebbe possibile sottrarre la cronologia e risalire al *token* ricevuto.

Mancanza di un *back channel* In assenza di un *backend* per un'applicazione web come *Angular*, i passaggi vengono eseguiti solo nel *front channel*, omettendo la parte di validazione e controllo aggiuntivo nei passaggi di invio e ricezione dati.

Browser instabile e insicuro Il *browser* è imprevedibile e pertanto è necessario coprire un maggiore numero problemi che possono avvenire al caricamento di una pagina (es. alterazione del codice sorgente di una pagina web, mancata garanzia di ricezione dati da parte di chi invia).

Per aumentare la sicurezza nell'*implicit flow*, il protocollo prevede un'estensione, denominata PKCE (*Proof Key for Code Exchange*, pronunciata "pixie"), attraverso cui è possibile sfruttare comunque l'*authorization code flow* nelle *Single Page Application* in *Javascript* e nelle applicazioni mobili. Tramite questa estensione è possibile aggiungere un controllo ulteriore all'inizio e alla fine del processo, che permette di generare un codice di autorizzazione segreto e verificarlo attraverso un opportuno algoritmo di *hash* prima di restituire l'*access token*.

Per quanto riguarda le altre principali vulnerabilità, *OAuth* non rimane comunque esente da altri problemi che possono accadere e che in passato sono accaduti anche ad aziende come *Twitter* e *Google*.

Salvataggio improprio delle chiavi segrete In un caso clamoroso del 2013, *Twitter* ha impropriamente inserito all'interno delle proprie applicazioni dei *secrets* con i quali è stato possibile impersonarsi come il *social network* nel processo di autorizzazione di OAuth⁴. Questo ha reso *OAuth* molto vulnerabile nel caso in cui si utilizzino sempre le stesse chiavi segrete; per evitare ciò è stato implementato *PKCE*, che genera ad ogni richiesta di autorizzazione nuove chiavi segrete.

Furto di *access tokens* Nel caso in cui non si controllino i *claims* che vengono messi a disposizione per un singolo *access token*, i rischi a livello di permessi possono essere molto alti. Le applicazioni web o mobili che utilizzano gli *access token* e che sono direttamente integrate come componenti di una mia architettura possono essere facilmente attaccate e pertanto le dovrei trattare come se fossero componenti di terze parti.

⁴ Twitter *OAuth API Keys leaked*, ThreatPost, 2013. URL: <https://threatpost.com/twitter-oauth-api-keys-leaked-030713/77597/>.

Authorization grant ambiguo Nel 2017 avvenne un caso di *phishing* nella piattaforma email di *Google*⁵, che causò l'invio di numerose email fasulle che richiedevano l'accesso a un documento *Google*. Il servizio, in realtà, aveva un nome uguale a quello originale e, a prima vista di un utente inesperto, poteva essere scambiato per quello autentico. Questo ha portato *Google* a rafforzare il quantitativo di informazioni da mostrare all'utente prima di concedere il consenso, dal momento che in caso di informazioni poco chiare un utente potrebbe consentire l'accesso ai propri dati a un'applicazione malevola.

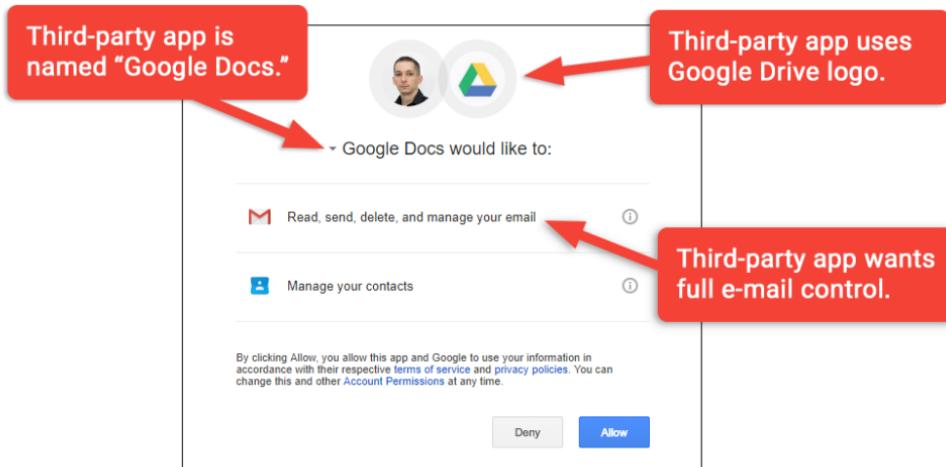


Figura 3.14: Consenso OAuth su Google nel 2017
Fonte: arstechnica.com

Nell'ambito del progetto, ho ritenuto essenziale la valutazione di questo protocollo per determinare le principali falle di sicurezza, così da interpretare a pieno i potenziali punti deboli che in fase di pianificazione avrei considerato insieme al *tutor* aziendale. L'analisi effettuata mi ha consentito dunque di decidere con maggiore cognizione di causa i benefici di sicurezza e il costo in termini di risorse per questo protocollo.

3.2.5 Applicabilità dei protocolli di sicurezza web

Dopo aver analizzato e studiato attentamente tutti i protocolli di interesse per l'azienda ho valutato una possibile applicabilità di ognuno di essi nel contesto del progetto

⁵ *Don't trust OAuth: Why the "Google Docs" worm was so convincing*, ArsTechnica, 2017. URL: <https://arstechnica.com/information-technology/2017/05/dont-trust-oauth-why-the-google-docs-worm-was-so-convincing/>.

SyncTrace. Per fare ciò ho analizzato i requisiti del progetto (vedasi sezione 2.4) e ho dato risposta alle seguenti domande che mi sono posto.

Quale componente richiede un’*autorizzazione* nella web app? Rifacendomi al protocollo *OAuth*, le principali autorizzazioni possono riguardare l’accesso a risorse protette che vengono interpellate attraverso richieste **CRUD** (*Create, Read, Update, Delete*) tramite protocollo HTTP(s). Dal momento che nella *web app* sono previste delle sezioni protette che contengono dati sensibili, il protocollo *OAuth* potrebbe entrare nel merito visto che mi permetterebbe di autorizzare con opportuni *scopes* le risorse a cui un utente può avere accesso.

Il servizio REST di *backend*, pertanto, dovrà essere costruito in modo che l’intestazione delle richieste in entrata e la relativa risposta seguano esattamente il documento di specifica *RFC 6749*⁶.

Quale componente richiede un’*autenticazione* nella web app? Riprendendo il protocollo *OpenID Connect*, invece, i principali casi d’uso possono riguardare il *login* e il *logout* nella *web app*, dal momento che, come riportato nei requisiti di progetto, dovranno essere presenti delle pagine protette attraverso cui un **utente autenticato** può interfacciarsi. Ciò mi ha portato a pensare che un’implementazione adeguata di questo protocollo avrebbe richiesto la realizzazione di un **servizio di identità**, grazie al quale sarebbe stato possibile ottenere un *ID token* associabile alla sessione di un utente nella *web app*. Un’applicazione di questo tipo comporterebbe sicuramente ampi vantaggi in termini di sicurezza, mentre a livello di costo di risorse il protocollo potrebbe essere facilmente adattabile a partire da *OAuth*.

A prima vista, ho ritenuto che la relazione tra i due protocolli per autorizzazione e autenticazione fosse particolarmente forte. L’implementazione del **servizio di identità** con *OpenID Connect*, inoltre, avrebbe ridotto di gran lunga il costo di implementazione per il protocollo *OAuth 2.0*, essendo stato costruito in un *layer* superiore.

Ragionando invece sull’aspetto architetturale e sull’effettivo *target* per cui esistono questi protocolli, ho considerato anche quanto segue:

- ***OAuth 2.0*** risulta essere particolarmente utile nel momento in cui si utilizza un **applicativo di terze parti** o un gran numero di microservizi interni alla mia architettura che vogliono avere accesso alle mie risorse protette;
- ***OpenID Connect***, in modo analogo, può essere utilizzato per lo stesso motivo di *OAuth*, con l’aggiunta di un **servizio di identità** che possa gestire tutta la parte di autenticazione su più applicativi.

Nel caso del progetto *SyncTrace*, rifacendomi ai componenti principali che sono stati pensati, ho ritenuto che l’unica applicazione che potrebbe sfruttare per intero un servizio di identità è la *web app*, ma una tale implementazione produrrebbe solo un **alto spreco di risorse**, dal momento che l’uso reale sarebbe limitato, appunto, a una sola applicazione. L’*app mobile*, infatti, non interagisce tramite autorizzazioni o autenticazioni particolari, e pertanto realizzare un’architettura con un componente così potente non porterebbe un maggiore valore aggiunto, se non in vista di una futura evoluzione del progetto che comporta l’aggiunta di nuovi componenti. L’applicazione web,

⁶ The *OAuth 2.0 Authorization Framework, Authorization Request and Response*, *RFC 6749*. URL: <https://tools.ietf.org/html/rfc6749#section-4.4.1>.

inoltre, essendo sviluppata come *Single Page Application* in *Javascript*, richiederebbe l'utilizzo del *implicit flow*, che risulta essere meno sicuro rispetto all'*authorization code flow*, visto che i *token* vengono inviati in chiaro sul *browser web*.

A partire da queste premesse, ho ritenuto opportuno sviluppare un *proof of concept* con il quale avrei mostrato i benefici di entrambi i protocolli e il modo con cui sarebbero stati effettivamente implementati per far fronte all'aspetto di autenticazione e autorizzazione.

Valutando altre alternative, ho spostato la mia attenzione anche sul *JSON Web Token*, da cui ho potuto capire se un servizio *ad hoc* e più personalizzabile sarebbe stato adeguato, visti i *framework* con i quali mi sarei dovuto interfacciare per la realizzazione della libreria.

3.3 Proof of Concept

Per partire bene con lo sviluppo della libreria, ho concordato insieme al proponente aziendale, l'ingegnere Fabio Pallaro, un insieme di componenti che sarei andato a mostrare in un applicativo di esempio da me sviluppato, il *proof of concept* appunto. Le motivazioni che mi hanno portato a spendere parte del mio tempo su questa attività sono riportate di seguito.

Approccio pratico alle tecnologie Un *proof of concept* mi permette di approcciarmi con le tecnologie studiate nella fase di analisi, provando a livello **pratico** ciascuna di esse. Questo mi facilita in fase di progettazione dal momento che con un *PoC* posso sapere con maggiore consapevolezza il costo in termini di risorse per l'implementazione di una certa tecnologia.

Baseline di prodotto Un *proof of concept* lo posso identificare come **baseline**, attraverso cui mi posso basare per iniziare la progettazione di dettaglio e lo sviluppo del mio prodotto, riutilizzando talvolta parte delle tecnologie e degli elementi implementati.

Approccio Agile Un *proof of concept* mi permette di mostrare agli *stakeholders* un **esempio** molto basilare del prodotto che voglio costruire e mi aiuta a identificare meglio gli obiettivi del prodotto finale, motivando così le scelte progettuali fin dal principio.

Attraverso la realizzazione del *proof of concept*, dunque, ho potuto mostrare a livello pratico il prodotto atteso dall'azienda, e questo mi ha permesso di comprendere meglio il contesto applicativo del progetto, identificando nel dettaglio le componenti e i servizi richiesti dalla mia libreria.

Partendo dal principio, il mio *proof of concept* si è basato su quanto segue:

- realizzare delle API di *mock* sotto lo standard *OpenAPI*, implementando la parte di *security* in alcune richieste *HTTP* messe a disposizione;
- mostrare la realizzazione di un servizio base per l'autenticazione all'interno di una *web app* in cui richiedere delle credenziali casuali, ritornando un *JSON Web Token* come risposta;

- implementare un servizio per la gestione della sessione, affinché venisse mantenuta direttamente nel *browser web* sfruttando il *web storage*;
- integrare una serie di pagine accessibili con e senza autenticazione che sfruttassero il protocollo *OAuth 2.0*, mostrando anche il funzionamento delle *API* in caso di errori del protocollo *HTTP*;
- realizzare una serie di maschere di una *web app* di esempio, utilizzando il *framework Angular* e implementando i relativi servizi e componenti con il linguaggio *TypeScript*;
- mostrare il funzionamento del protocollo *OpenID Connect* con una pagina di esempio che sfrutta un servizio esterno di terze parti e con il *token* di ritorno per mantenere l'accesso.

Sviluppando per intero tutti i punti sopraelencati, ho potuto mostrare in modo esaustivo le conoscenze apprese e ho identificato con maggior accuratezza il funzionamento dei singoli componenti. In particolare, riporto di seguito alcune delle procedure più rilevanti del mio *proof of concept* che mostrano il funzionamento dei protocolli.

Partendo dal protocollo *OAuth 2.0*, ho voluto testare una pagina privata che generasse una lista di aziende a seguito di una chiamata a un server di *mock* per le *API* sfruttando lo strumento *Prism*, il quale genera delle risposte dinamiche a *runtime*, validando nello specifico ciascuna richiesta in entrata e riportando eventuali errori *HTTP* se specificati (figura 3.15).

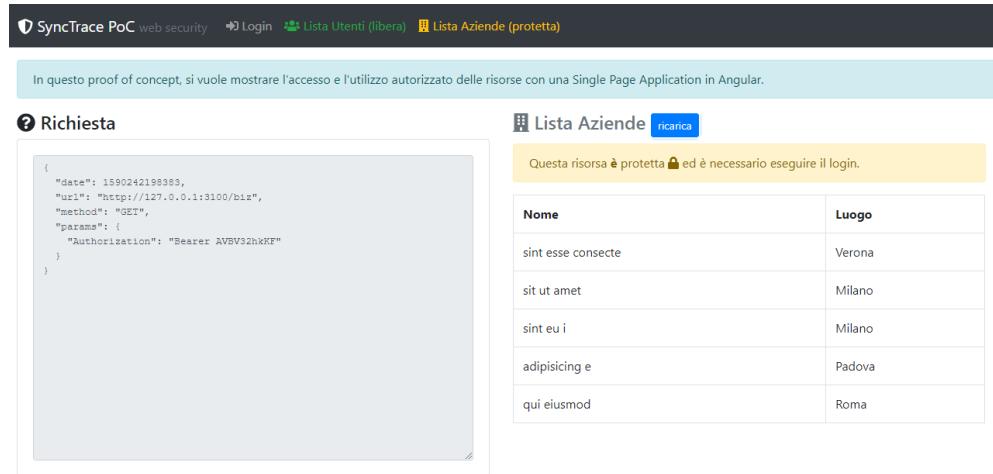


Figura 3.15: Proof of concept, pagina protetta che sfrutta il protocollo OAuth 2.0

Nel caso in cui una richiesta *HTTP* che utilizza il protocollo *OAuth 2.0* fosse eseguita senza l'utilizzo di un *token* di autorizzazione, rappresentabile da una stringa alfanumerica o, se necessario, da un *JWT*, l'applicativo restituisce un errore dalle *API*,

nello specifico un errore *HTTP 401 Unauthorized*, come specificato nel documento RFC 7231⁷.

The screenshot shows a web application interface. At the top, there is a header bar with a back arrow and the text "Risposta". Below the header, a modal window is displayed. The modal has a yellow header bar with the text "Lista Aziende" and "ricerca". The main body of the modal contains the following text:

```
{
  "headers": {
    "normalizedNames": {},
    "lazyUpdate": null
  },
  "status": 401,
  "statusText": "Unauthorized",
  "url": "http://127.0.0.1:3100/biz",
  "ok": false,
  "name": "HttpErrorResponse",
  "message": "Http failure response for http://127.0.0.1:3100/biz: 401 Unauthorized",
  "error": {
    "type": "https://stoplight.io/prism/errors#UNAUTHORIZED",
    "title": "Invalid security scheme used",
    "status": 401,
    "detail": "",
    "headers": {
      "WWW-Authenticate": "OAuth2"
    }
  }
}
```

Figura 3.16: Proof of concept, errore per mancata autorizzazione sfruttando il protocollo OAuth 2.0

Per quanto riguarda il protocollo *OpenID Connect*, invece, ho utilizzato un servizio esterno che mi ha permesso di simulare le richieste verso un *Authorization Server* a partire dalla *web app*, gestendo così le relative richieste di ritorno e i relativi controlli in modo standard. Come è possibile notare dalla figura 3.17, premendo il bottone *Esegui l'accesso al sito*, viene eseguito un *redirect* a un servizio esterno di identità, preparando la richiesta come è possibile vedere nel *box* a sinistra.

The screenshot shows a web application interface. At the top, there is a header bar with several items: "SyncTrace PoC web security", "OpenID Connect", "Login", "Lista Utenti (libera)", and "Lista Aziende (protetta)". Below the header, a modal window is displayed. The modal has a yellow header bar with the text "Richiesta". The main body of the modal contains the following text:

La richiesta mostra la pagina che verrà aperta una volta cliccato sul bottone. Come è possibile vedere, gli **scope** richiesti dal client sono **openid e profile**. Di questi verrà richiesta autorizzazione.

Below this text is a code snippet:

```
https://phantauth.net/auth/authorize?client_id=poc-web-security&response_type=token&id_token_scope=openid&profile&state=abc&nonce=123&redirect_uri=http://localhost:4200/cide
```

To the right of the "Richiesta" box, there is another box titled "Login OpenID connect (PhantAuth.net)". This box contains the following text:

Accesso tramite OpenID Connect con PhantAuth.net come issuer endpoint.

Passaggi per il login

- Cliccare sul bottone verde in basso. Avverrà un redirect su una pagina di login esterna.
- Cliccare sul bottone **login** che comparirà.
- Autorizzare gli **scope** richiesti.
- La pagina riporterà al login con l'access token e l'id token.

At the bottom of this box is a green button labeled "Esegui l'accesso al sito".

Proof of Concept - Web security - Mariano Sciacco

Figura 3.17: Proof of concept, pagina del client che vuole eseguire l'accesso alla web app usando OpenID Connect

Una volta eseguito il *redirect*, l'utente deve inserire le proprie credenziali, che in questo caso sono generate casualmente essendo un servizio di identità di prova (figura

⁷ Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Response Status Codes, RFC 7231. URL: <https://tools.ietf.org/html/rfc7231#section-6>.

3.18). Successivamente viene richiesto di autorizzare l'applicazione all'uso dei dati dell'utente, a cui può concedere accesso totale o parziale, in base alle proprie esigenze.

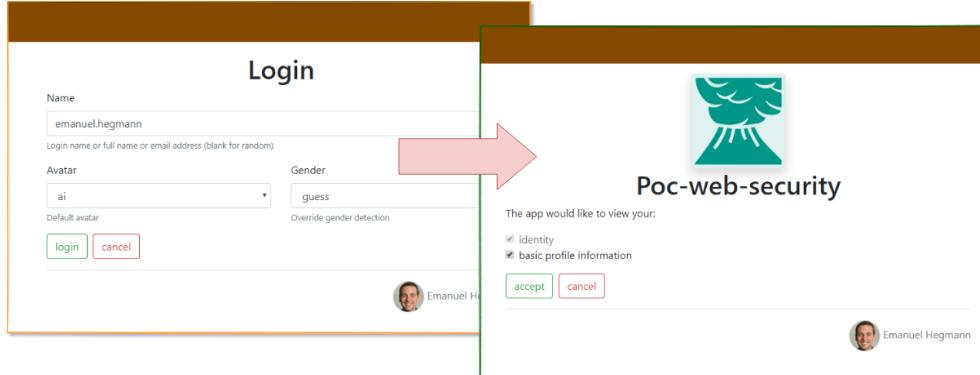


Figura 3.18: Proof of concept, servizio di identità esterno per l'accesso alla web app tramite OpenID Connect

In conclusione, se il processo è andato a buon fine viene eseguito un ultimo *redirect* alla *web app* con in allegato un *access token* per l'autorizzazione e un *ID token* (in formato *JWT*) per la sessione dell'utente (figura 3.19).

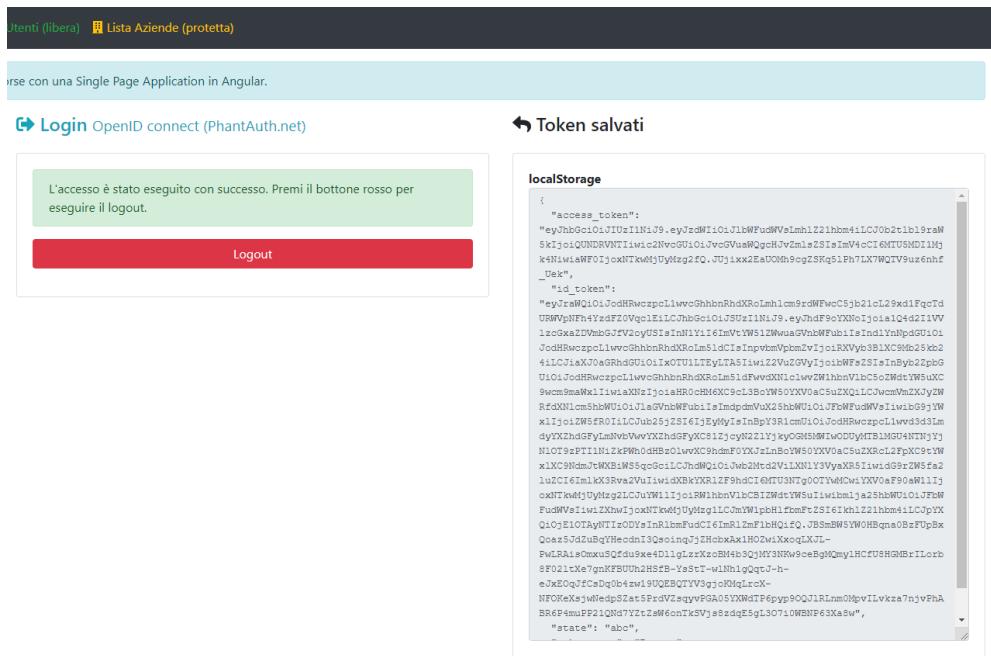


Figura 3.19: Proof of concept, redirect e accesso finale all'applicazione web usando OpenID Connect

Ho realizzato il *proof of concept* anche per provare le tecnologie di sviluppo principali come il *framework Angular*, il linguaggio *TypeScript* e *Javascript*, il protocollo *HTTP* e le politiche di condivisione e richiesta di risorse gestite attraverso il *Cross-Origin*

Resource Sharing, che permette di bloccare o autorizzare richieste *HTTP* tramite *Javascript* a seguito di una richiesta di un servizio *backend* (es. *API REST*) tenuto su un *URL* (o un server) differente.

A seguito della realizzazione del *proof of concept*, ho parlato con il *tutor* aziendale, il quale è rimasto molto soddisfatto dell'analisi svolta, e, insieme, ci siamo confrontati in merito alla completezza delle tecnologie riportate e alla loro effettiva implementazione nel mio progetto. In fase di pianificazione mi sono trovato così più agevolato nel riuscire a consolidare i requisiti più importanti e i casi d'uso della libreria, considerando anche il tempo a mia disposizione e le necessità del progetto.

3.4 Progettazione

Per la progettazione della libreria, ho fatto uso del *proof of concept* per riprendere parzialmente le tecnologie che avevo già implementato, selezionando solo quelle più adeguate che sarebbero state essenziali per mettere in sicurezza l'applicazione *web*. La progettazione, pertanto, è stata per me fondamentale per determinare nel dettaglio le funzionalità della libreria, così da rimanere comunque entro le tempistiche prefissate dal progetto di stage.

3.4.1 Identificazione delle tecnologie

Partendo dal *proof of concept*, ho coinvolto il *tutor* aziendale in merito alla scelta dei protocolli e delle tecnologie da integrare nella libreria. Per quanto riguarda i protocolli, ho portato due opzioni che sarebbero state perseguitibili per la realizzazione dei servizi di autorizzazione e autenticazione nella *web app*:

- **Servizio di identità:** con questa prima opzione, ho proposto di integrare i due protocolli studiati, *OAuth 2.0* e *OpenID Connect*, così da realizzare un applicativo indipendente su cui io mi sarei appoggiato per eseguire richieste di autorizzazione e autenticazione dalla *web app*. Questo, ovviamente, comporta alti costi di realizzazione, ma favorisce sicuramente un maggiore grado di sicurezza e di estensibilità nel caso di una futura evoluzione del progetto (es. nuovo componente);
- **Servizio di sicurezza *ad hoc*:** con questa seconda opzione, ho avanzato l'idea di realizzare un servizio di sicurezza *ad hoc* che potesse pescare parte delle qualità di entrambi i protocolli studiati, ma che abbattesse i costi a livello di progettazione e implementazione, rimanendo comunque compatibile e riutilizzabile anche per altri progetti di *Angular*. Inoltre, a livello di sviluppo *backend* l'implementazione sarebbe stata nettamente più semplice e adeguata per gli attuali componenti del progetto, limitando però il livello di sicurezza a un protocollo non standard.

Su discussione e approvazione del proponente aziendale, ho scelto di realizzare il **servizio di sicurezza *ad hoc***, dal momento che sarebbe stato molto più attinente all'architettura attuale del progetto. Inoltre, in vista di nuove possibili estensioni che comportano l'aggiunta di altri componenti, l'azienda avrebbe preferito cercare altre soluzioni *ad hoc* in termini di sicurezza. A livello di risorse, inoltre, la seconda opzione ha permesso di abbattere i costi di realizzazione, riducendo anche la complessità di implementazione che avrebbe comportato la prima opzione. Per quanto riguarda il grado di sicurezza, ho garantito questa qualità sulla base del mio studio realizzato in fase di analisi, che pone particolare attenzione alle *best practice* di entrambi i protocolli.

	S. DI IDENTITÀ	S. DI SICUREZZA
Costo di realizzazione	Alto	Basso
Adotta uno standard	Sì	No
Grado di sicurezza	Alto	Medio
Riutilizzabile su altri progetti	No	Sì
Estensibile su nuovi componenti	Sì	No
Implementazione	Lunga e complessa	Ridotta e semplice
Componente più coinvolto	<i>API REST</i>	<i>Web application</i>

Tabella 3.2: Tabella riassuntiva sulla scelta dei servizi da implementare nella libreria

A partire da questa scelta, ho deciso di utilizzare tutto quello che mi sarebbe servito per l'implementazione completa della libreria, sfruttando gran parte delle tecnologie che avevo studiato e trattato precedentemente (sezione 3.1) e di cui l'azienda stessa ne ha già fatto grande uso in molti altri progetti interni:

- **JSON Web Token**, per la realizzazione e gestione delle sessioni utente all'interno della *web app*, riutilizzando quanto fatto nel *proof of concept*;
- **framework Angular (TypeScript)**, per l'implementazione della libreria con tutte le singole componenti richieste;
- **Stoplight Studio (OpenAPI 3.0)**, per la definizione delle API richieste dalla libreria, utilizzate poi anche nei test di integrazione;
- **Prism**, per l'esecuzione di un *server* di *mock* delle *API* con richieste dinamiche e auto-generate in base al documento *OpenAPI* realizzato su *Stoplight Studio*;
- **Jasmine**, per la realizzazione dei casi di test integrati direttamente in *Angular*, la misurazione del *code coverage* e la verifica dell'applicativo;
- **Node package manager (npm)**, per la gestione delle dipendenze, l'avvio dei casi di test, la costruzione (*build*) della libreria e il rilascio.

3.4.2 Requisiti della libreria

Per poter avere una visione d'insieme dei requisiti, ho realizzato prima di tutto una serie di **diagrammi di casi d'uso** della libreria in cui un utente finale avrebbe interagito ad alto livello. Attraverso i casi d'uso ho potuto sottolineare i principali casi di interazione di un utente nella *web app* di cui il mio applicativo si sarebbe occupato dal punto di vista di sicurezza. Pertanto, non ho basato i casi d'uso solamente sugli scopi iniziali della libreria, come riportato nella sezione 2.5, ma anche sul *proof of concept* realizzato a seguito della fase di analisi.

Ho identificato e documentato ciascun caso d'uso in modo univoco, sfruttando il seguente formato:

$$\text{UC}[\alpha].[\beta]$$

dove:

- α : identifica il codice del caso d'uso con un numero progressivo che parte da 1;

- β : opzionale, identifica il codice figlio del caso d'uso padre (α) con un numero progressivo che parte da 1.

Per ciascun caso d'uso ho riportate anche le seguenti informazioni:

- **diagramma UML**: diagramma del caso d'uso in formato *UML 2.0* (Unified Modeling Language), realizzato con lo strumento *StarUML*;
- **attori primari**: attori principali del caso d'uso, che interagiscono direttamente con il sistema;
- **attori secondari**: attori secondari del caso d'uso, che si identificano generalmente con applicazioni di terze parti e che interagiscono con il sistema;
- **descrizione**: breve descrizione del caso d'uso;
- **estensioni**: eventuali estensioni coinvolte (es. errori di compilazione di un *form*);
- **inclusioni**: eventuali inclusioni coinvolte;
- **precondizione**: condizioni che devono essere soddisfatte perché si verifichino gli eventi del caso d'uso;
- **post-condizione**: condizioni che devono essere soddisfatte dopo il verificarsi degli eventi del caso d'uso;
- **scenario principale**: flusso degli eventi, in forma di elenco numerato, con eventuale riferimento a ulteriori casi d'uso.

Per questo progetto, ho limitato il numero dei casi d'uso alle sole interazioni principali che la libreria dovrà prevedere, e i risultati di analisi sono i seguenti:

DATO	VALORE
Nr. di casi d'uso padre	4
Nr. di casi d'uso totali	8
Nr. di attori	2

Tabella 3.3: Tabella riassuntiva dei casi d'uso identificati per il progetto

Dal momento che la mia libreria sarebbe stata componente essenziale della *business-logic* dell'applicativo web, ho visto i casi d'uso come un valore aggiunto, che mi ha permesso di indirizzare meglio l'utilizzo della libreria. Tuttavia, lo studio più rilevante ha riguardato principalmente le interazioni tra i componenti della libreria, documentato tramite opportuni diagrammi di sequenza.

A seguito delle scelte sulle tecnologie da implementare e della definizione dei casi d'uso, ho stilato una lista dei requisiti ordinata e organizzata, che classificasse ciascun requisito per importanza e per tipologia. In particolare, ho usato il seguente formato:

$$R[\alpha] - [\beta] - [\gamma]$$

dove:

- α : assume, a seconda del grado di priorità, i valori:
 - **A**: obbligatorio, cioè strettamente necessario e non negoziabile per la realizzazione del prodotto;
 - **B**: desiderabile, cioè non strettamente necessario, ma dal discreto valore aggiunto;
 - **C**: opzionale, cioè non necessario e contrattabile in corso d'opera;
- β : a seconda del tipo di requisito, assume i valori:
 - **F**: funzionale, che identifica una funzionalità del prodotto finale;
 - **P**: prestazionale, che identifica una caratteristica in termini di prestazioni del prodotto finale;
 - **Q**: qualitativo, che identifica una qualità che il prodotto finale deve possedere;
 - **V**: vincolo, che identifica un vincolo imposto dal progetto;
- γ : numero progressivo che parte da 1 per contraddistinguere il requisito indipendentemente dalla priorità e dalla tipologia.

In totale, ho riportato **28 requisiti**, suddivisi nel seguente modo:

DATO	VALORE
Nr. di requisiti funzionali	23
Nr. di requisiti prestazionali	0
Nr. di requisiti qualitativi	2
Nr. di requisiti di vincolo	3
Numero totale di requisiti	28

Tabella 3.4: Tabella riassuntiva dei requisiti identificati per il progetto

I **requisiti funzionali** hanno riportato tutte le funzionalità previste per la libreria, rifacendosi anche a quanto riportato nello scopo dello stage (sezione 2.5). Alcuni di questi requisiti, ad esempio, sono stati:

ID	DESCRIZIONE
RA-F-4	La libreria deve permettere di autenticare un utente con le credenziali corrette.
RA-F-7	La libreria deve disconnettere l'utente se la sessione risulta scaduta.
RA-F-13	La libreria deve gestire gli <i>header</i> da inviare al <i>backend</i> .
RA-F-16	La libreria deve permettere di eseguire e gestire richieste <i>GET</i> .
RA-F-17	La libreria deve permettere di eseguire e gestire richieste <i>POST</i> .

Tabella 3.5: Estratto di alcuni requisiti funzionali del progetto

Non ho definito dei **requisiti prestazionali** dal momento che in fase di analisi non ho ritenuto necessario garantire alcun tipo di prestazione particolare per la libreria. Per quanto riguarda i **requisiti qualitativi**, ho provveduto a inserire come requisito l'utilizzo di un sistema di versionamento (*GitLab*) e l'implementazione dei casi di test, così che il prodotto venisse versionato e verificato nel corso dello sviluppo, riducendo al minimo il numero di *bug*. Infine, per quanto concerne i **requisiti di vincolo**, mi sono rifatto ai principali obblighi di progetto che mi sono stati imposti dal *tutor aziendale*, specialmente in termini di *framework* e di lingua del prodotto.

3.4.3 Soluzioni progettuali

Ho basato le principali soluzioni progettuali della libreria sull'interpretazione dei requisiti di progetto e sulle *best practice* del *framework* di riferimento per lo sviluppo. A partire dalle mie conoscenze e dallo studio di *Angular*, ho discusso alcune soluzioni insieme al *tutor aziendale* per determinare le scelte di progettazione che più si adeguassero al mio prodotto.

Soluzione di sicurezza *ad hoc* Come spiegato nella sezione 3.4.1, le interazioni della libreria sarebbero state realizzate *ad hoc* con il servizio *backend* di *API REST*, senza la necessità di implementare i protocolli per autorizzazione e autenticazione studiati, ma prendendo semplicemente spunto da essi;

Rilascio, integrazione e riutilizzo della libreria Nel corso dello sviluppo, mi sono impegnato a realizzare una libreria che fosse il più possibile riutilizzabile e indipendente dall'applicativo web. Per questo motivo, ho creato un progetto separato rispetto alla *web app*, che potesse poi essere integrato come **package** tramite l'utilizzo di *node package manager*, il gestore di dipendenze. Ciò è stato in buona parte possibile e ha agevolato di gran lunga il processo di **integrazione** nella libreria, dal momento che i compilati della mia libreria potevano essere scaricati e integrati come **dipendenza** nel progetto principale, e con un semplice comando (`npm update <libreria>`) eventuali aggiornamenti potevano essere applicati anche dai colleghi che si occupavano della *web app*, evitando così possibili conflitti;

Scelta dei Web Storage Attraverso i *web storage*, è possibile salvare dei dati nel *browser web* in modo molto simile ai *cookie*. Dopo aver condotto uno studio approfondito su queste tecnologie, ho deciso di mettere a disposizione una scelta integrata nella configurazione che potesse essere cambiata in base alle esigenze. In particolare:

- **localStorage**: i dati vengono salvati in una *cache* locale che permane nel *browser* fintanto che non viene manualmente cancellata dall'applicativo web o dall'utente; questo permette quindi di mantenere le sessioni degli utenti attive, rendendo però quest'ultimi più esposti a potenziali attacchi che comportano il furto dei dati contenuti nello *storage*;
- **sessionStorage**: il funzionamento è molto simile al *localStorage*, con l'unica differenza che una volta che il *browser* (nello specifico un *tab* del *browser*) viene chiuso, i dati presenti vengono distrutti, obbligando l'utente a più accessi, ma riducendo il numero di potenziali attacchi, visto che i dati permangono per molto meno.

Observables over Promises Dal punto di vista implementativo, ho preferito l'utilizzo degli *Observables*, essendo un'evoluzione sostanziale delle *Promise* nell'ambito delle chiamate asincrone in *Angular*. Una *Promise* permette di gestire un singolo evento quando un'operazione asincrona va in successo o in fallimento, mentre un *Observable* permette di gestire zero o più eventi, con un funzionamento analogo agli *Stream* in *Java*. Questo porta maggiori vantaggi anche dal punto di vista gestionale degli eventi, visto che questi possono essere cancellati, riducendo così il costo di eventuali chiamate inutili.

Per quanto concerne i *design pattern*, il *framework Angular* fa uso di moltissime soluzioni native che integrano alcune delle soluzioni fondamentali dettate dalla *Gang of Four*, come riportato nel loro libro *Design Patterns*⁸. Alcune delle soluzioni principali che ho integrato tramite il *framework Angular* sono riportate di seguito:

- **Singleton:** il *singleton* è uno dei *design pattern* più semplici che viene utilizzato per attivare una e una sola istanza di classe. Il *singleton*, pertanto, costruisce un oggetto di classe una sola volta e lo riutilizza per tutte le operazioni principali. Questo *design pattern* lo ho ritrovato nei *Service* di *Angular*, ossia i servizi che vengono utilizzati per la *business-logic* dell'applicazione, e nel mio caso è stato ampiamente utilizzato per realizzare gran parte dei servizi di gestione delle sessioni e gestione delle chiamate *HTTP* della libreria.
- **Decorator:** il *decorator* definisce una serie di nuove funzionalità che possono essere aggiunte a un oggetto esistente tramite l'utilizzo di un contenitore. Attraverso il contenitore è possibile costruire l'oggetto e usare direttamente tutti i metodi che mette a disposizione, senza bisogno di legami di dipendenza troppo forti. In *Angular*, ho ritrovato molti *decorator* tra cui *Injectable()*, che mi ha consentito di iniettare delle dipendenze in alcune classi del progetto.
- **Dependency Injection:** attraverso la *dependency injection*, le dipendenze di una classe vengono aggiunte direttamente in fase di costruzione di un oggetto. Questo è stato particolarmente enfatizzato in *Angular* per la costruzione dei servizi e dei componenti, dal momento che nei costruttori è richiesto di dichiarare i singoli servizi di cui si fa uso, facilitandone così l'individuazione in fase di sviluppo.
- **Pattern Model-View-ViewModel (MVVM):** *Angular* fa uso di questo *pattern* per la comunicazione tra il modello dati (*model*) e la vista (*view*), garantendo così maggiore separazione delle classi e dei componenti. In particolare *MVVM*, rispetto al *design Model-View-Controller*, aumenta le funzionalità della *View* in favore di un ruolo più attivo. A differenza del *Controller*, il *ViewModel* è un *Model* esteso con funzionalità per la manipolazione dei dati e per l'interazione con la *View*.

Durante la progettazione delle classi, ho utilizzato i seguenti *design pattern* all'interno della mia libreria, così che potessero facilitarmi nello sviluppo:

Pattern Facade Il *facade* permette di accedere a sottosistemi che espongono interfacce più complesse e, talvolta, molto diverse tra di loro, fornendo allo sviluppatore un'interfaccia più semplice come punto di accesso, senza limitare comunque l'accesso

⁸E. Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

alle classi del sistema. Implementando questo *pattern*, ho fornito alla libreria un singolo servizio in cui esporre i principali metodi per utilizzare in modo completo le funzionalità definite nei requisiti. Questo ha permesso di semplificare per me la progettazione delle singole componenti, separando i *package* nelle funzionalità comuni e creando anche per loro dei *facade* che potessero essere a loro volta utilizzabili anche singolarmente.

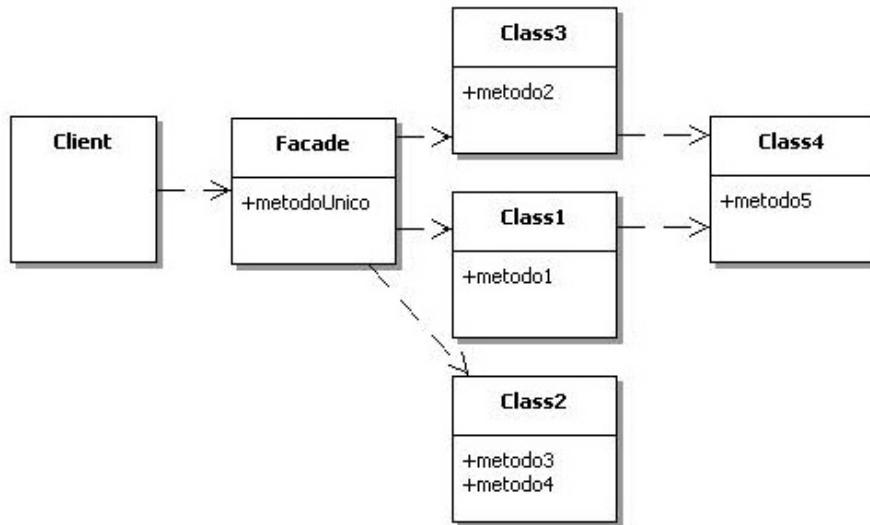


Figura 3.20: Diagramma per il design pattern Facade

Fonte: wikipedia.org

Pattern Command Il *command pattern* permette di realizzare una serie di classi, identificate come *comandi*, che derivano tutti da un’interfaccia comune e ciò favorisce l’isolamento del codice anche complesso, nonché l’estensibilità futura. I comandi vengono chiamati da un *invoker* che li esegue senza bisogno di conoscere l’implementazione di ciascuno di essi. Questo *pattern* mi è stato particolarmente utile nella libreria per definire una serie di chiamate *HTTP* con implementazione differente, ma con comportamento a livello di chiamata molto simile.

3.4.4 Architettura della libreria

Ho composto l’architettura della mia libreria con una serie di *package* che sono stati utili a separare per funzionalità le singole componenti. Ho progettato ciascun *package* con lo scopo di occuparsi di una singola mansione, così come per le classi implementate al loro interno. Per fare ciò mi sono posto l’obiettivo di separare il più possibile le componenti in modo che ognuna potesse funzionare in modo indipendente dall’altra, rifacendomi così ai principi *SOLID*⁹ per la programmazione a oggetti.

⁹ S.O.L.I.D: The First 5 Principles of Object Oriented Design. URL: <https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>.

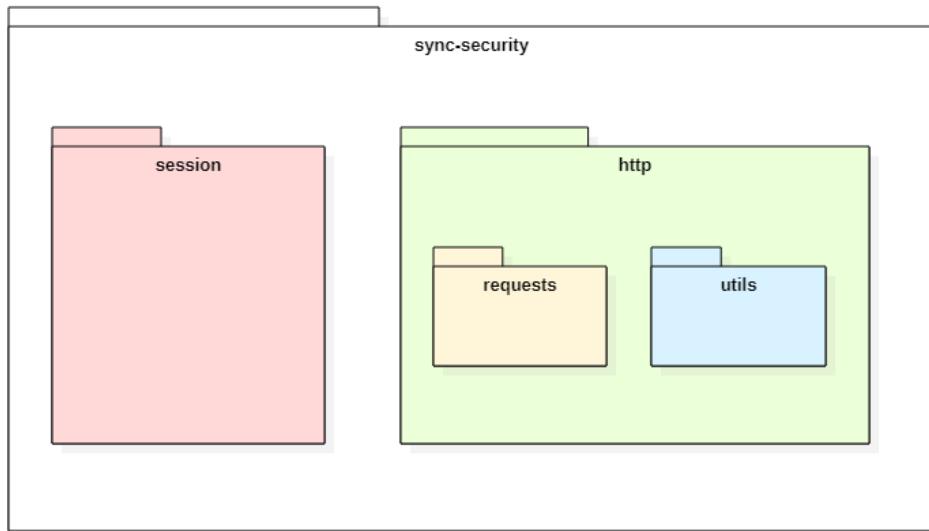


Figura 3.21: Diagramma dei package della libreria di progetto

A livello architetturale, ho suddiviso la libreria in 3 *package* principali, attraverso cui ho gestito tutte le principali funzionalità richieste dal progetto.

Package sync-security In questo *package* sono presenti tutte le classi che ho realizzato per la libreria, in cui ho esposto un servizio (*SyncSecurity*) con il *design pattern facade* con cui è possibile fruire di tutte le operazioni principali della libreria; tra i vari metodi, ho definito anche delle funzioni per l'accesso, il *logout*, le richieste *CRUD* via *HTTP* e la gestione della sessione.

Package sync-security.session Attraverso il *package session*, ho definito tutte le classi e le funzioni per gestire le sessioni dell'utente all'interno della *web app*, manipolando il *web storage* del *browser* e il *JSON Web Token* ricevuto a seguito dell'autenticazione.

Package sync-security.http Nel *package http* ho voluto integrare tutte le operazioni di invio asincrono delle richieste *HTTP*, gestendo in modo particolare le intestazioni con l'autorizzazione per l'accesso alle risorse delle *API REST*. Oltre a ciò, ho definito anche una serie di richieste che possono essere facilmente estese grazie all'uso del *design pattern command* e un insieme di serializzatori per preparare i contenuti da inviare con il protocollo *HTTP*.

Rifacendomi quindi al *proof of concept*, ho proceduto con restaurare parti dei diagrammi di sequenza che avevo realizzato per i protocolli *OAuth* e *OIDC*, riportando le principali interazioni tra le varie classi, che illustravano la richiesta di una risorsa tramite *HTTP* e l'accesso dell'utente all'interno dell'applicativo web. A titolo esemplificativo, riporto solamente alcuni dei diagrammi ad alto livello, che rappresentano autenticazione e autorizzazione.

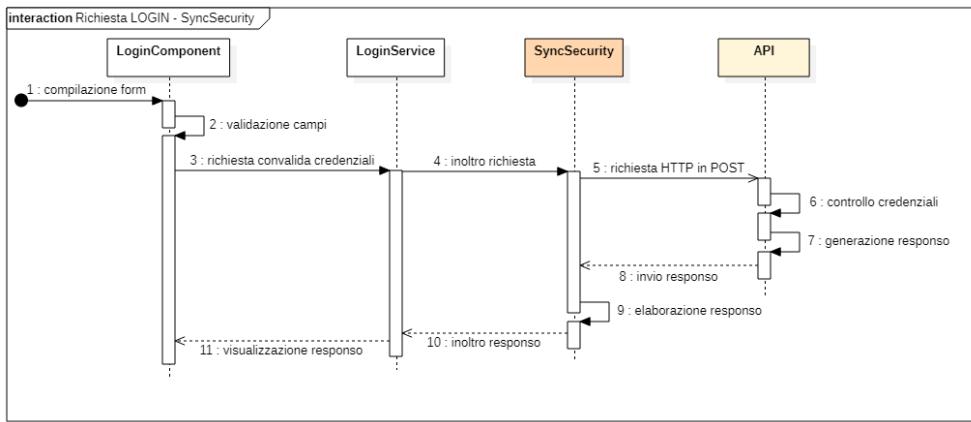


Figura 3.22: Diagramma di sequenza per il processo di autenticazione utilizzando la libreria di progetto

Partendo con l'**autenticazione**, ho cercato di rendere il più indipendente possibile questo processo, così da inserirsi a metà tra il servizio della *web app* e il servizio *backend* in *API REST*. Come riportato in figura 3.22, l'utente invia delle credenziali che vengono validate e gestite dal *LoginService*. Questo le inoltra alla mia libreria, eseguendo una chiamata alla funzione di *login*, che gestisce per intero la richiesta e interagisce con il servizio *backend* esterno.

Una volta che le credenziali vengono validate dalle *API*, viene ricevuto un risponso che può contenere un *JSON Web Token*, che sarà poi utilizzato per la sessione. Il risponso verrà inoltrato anche al servizio *LoginService* che potrà leggere il messaggio contenuto e riportarlo anche al *LoginComponent*.

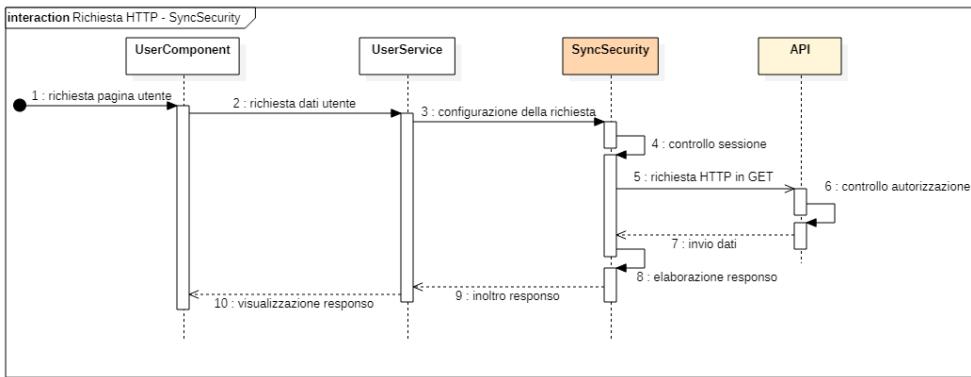


Figura 3.23: Diagramma di sequenza per la richiesta di una risorsa via HTTP utilizzando un servizio della libreria di progetto

Per il processo di **autorizzazione**, invece, ho simulato la richiesta a una risorsa che riguardasse una pagina profilo di un utente. Come è possibile vedere dalla figura 3.23, l'utente richiede di visualizzare una pagina che viene gestita dal servizio *UserService*, il quale, dopo aver configurato una richiesta con tutti gli elementi richiesti, sfrutterà una mia funzione della mia libreria che gestirà per intero tutta la richiesta HTTP verso

il servizio esterno di *API*.

Successivamente, la mia libreria elaborerà e gestirà il responso, fornendo poi l'esito della richiesta al servizio chiamato, con i dati formattati in allegato.

3.5 Codifica e verifica del software

Nella fase di sviluppo della libreria, ho integrato per una buona parte del tempo l'attività di verifica, alimentando in parallelo una corretta implementazione delle classi progettate nella fase precedente. Pertanto, lo sviluppo ha comportato l'impiego di quasi due settimane piene di lavoro, che mi hanno condotto alla realizzazione di tutte le classi fondamentali della libreria. A seguito della codifica e verifica, la libreria è stata validata e sono stati perfezionati i diagrammi realizzati fino ad allora.

Ho eseguito la codifica della libreria a partire da un primo **applicativo di esempio**, realizzato sulla base del *proof of concept*, e ciò mi ha aiutato per avere un riferimento di tutte le funzionalità essenziali della libreria. Con questo applicativo, ho potuto anche mostrare costantemente al mio *tutor* aziendale gli **avanzamenti** della libreria, conciliando anche l'attività di verifica in parallelo.

Come è possibile vedere in figura 3.24, l'applicazione di esempio mostra una funzionalità di *login* attraverso cui ho reso possibile **autenticarsi** per avviare una sessione utente, inserendo delle apposite credenziali casuali.

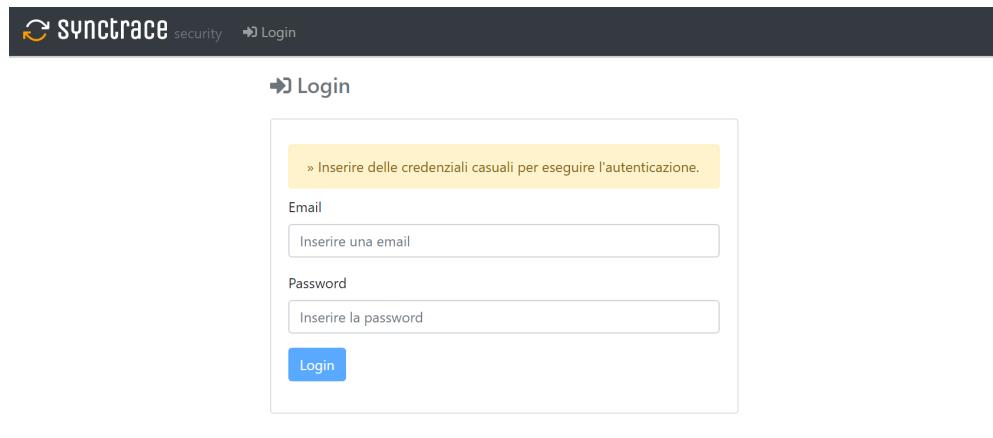


Figura 3.24: Pagina login dell'applicazione di esempio per mostrare il funzionamento della libreria

Nella figura 3.25, ho mostrato il funzionamento della parte di **autorizzazione** in cui ho reso possibile provare l'invio di richieste *HTTP* con un apposito *header* e parti della richiesta personalizzate. I bottoni a sinistra, infatti, hanno permesso di cambiare dei parametri della richiesta in uscita verso le *API*.

The screenshot shows the 'Dashboard' page of the Synctrace application. At the top, there's a header with the Synctrace logo, a 'Login' link, and a 'Dashboard' link. Below the header, the main content area has a title 'Lista infetti e guariti' and a message stating there are 4 records. A table lists four individuals: Massimo (Catania, 42 anni, Positivo), Giulia (Asiago, 40 anni, Positivo), Anna (Asiago, 42 anni, Negativo (guarito)), and another Massimo (Catania, 40 anni, Positivo). On the left side, there's a sidebar with 'Informazioni' (logged in as Mario Rossi) and 'Opzioni' (with buttons for 'Visualizza lista completa', 'Visualizza infetti guariti', and 'Visualizza infetti positivi'). At the bottom right, it says 'SyncSecurity - Mariano Sciacco'.

Nome	Luogo	Età	Status corrente
Massimo	Catania	42 anni	Positivo
Giulia	Asiago	40 anni	Positivo
Anna	Asiago	42 anni	Negativo (guarito)
Massimo	Catania	40 anni	Positivo

Figura 3.25: Pagina protetta dell'applicazione di esempio per mostrare il funzionamento della libreria

Per quanto concerne la parte tecnica del progetto *Angular*, ho fatto uso di ***AngularCLI*** per l'aggiunta automatica di nuove classi al mio progetto. Ciascun file generato veniva automaticamente posizionato nella cartella di lavoro (figura 3.26) e conteneva del codice *boilerplate* pronto all'uso.

Di seguito, riporto alcuni dei principali elementi di cui ho avuto bisogno nel corso dello sviluppo, che mi hanno permesso di realizzare tutta la parte di *business-logic* della libreria, nonché la parte di struttura dati.

Service Un *service* permette di definire la *business-logic* del progetto e, generalmente, si può identificare come modello (*model*) in cui è possibile svolgere le principali operazioni che verranno poi riutilizzate da altri componenti. Proprio poiché vengono usati da altre classi, i *service* vengono inferiti nei costruttori come singole istanze di un oggetto tramite il *design pattern singleton*.

Component Un *component* identifica letteralmente un componente che viene direttamente collegato con una *view* del progetto. Il *component*, pertanto, rappresenta per intero un *ViewModel* nel *pattern MVVM*, dal momento che può manipolare i dati dai *Service* e interagire direttamente con la *View*.

Interface Un *interface* permette di definire un'interfaccia attraverso cui eseguire l'implementazione con una classe concreta. Questo elemento è molto utile per definire strutture dati che verranno poi riutilizzate nel progetto, a partire, ad esempio, dai dati forniti attraverso chiamate alle *API*.

Class Un *class* rappresenta letteralmente una classe del progetto, attraverso cui è possibile implementare interfacce o usarle in modo generico come oggetti per comporre più funzionalità per la *business-logic* del programma.

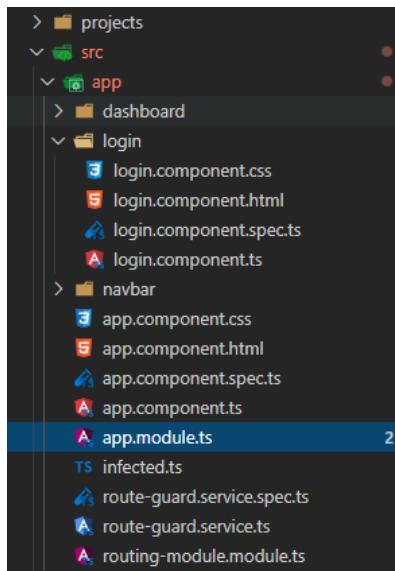


Figura 3.26: Disposizione delle cartelle nell'applicazione di esempio del progetto Angular

A partire dalla conoscenza degli elementi richiesti dal progetto, l'iter di sviluppo ha seguito le seguenti "tappe", che più formalmente potremmo chiamare *baseline*, essendo state incrementi verificati del mio applicativo:

1. **Firma dei metodi del servizio SyncSecurity:** a partire dal servizio principale ho realizzato la firma dei metodi che avrei dovuto poi implementare una volta che le funzionalità nei *package* sarebbero state inserite;
2. **Implementazione del package http:** per prima cosa, ho voluto concentrarmi nello sviluppo del *package http* essendo fondamentale per la parte di autorizzazione e interazione con il servizio *backend*. A corredo, ho realizzato anche alcuni serializzatori in un *sub-package http.utils*, così da poter far funzionare a pieno le richieste *HTTP*;
3. **Implementazione del package session:** ho implementato il *package session* a seguito del *package http*, così da integrare tutte le funzionalità relative alla gestione delle sessioni e alla manipolazione dei *tokens*, in particolare di *JSON Web Token*;
4. **Definizione dei metodi del servizio SyncSecurity:** a conclusione dello sviluppo dei package, ho implementato e definito il servizio principale, *SyncSecurity*, con il quale ho potuto testare poi l'effettivo funzionamento anche dall'applicazione di esempio.
5. **Predisposizione dei comandi di build e rilascio del prodotto:** dopo aver portato a termine lo sviluppo di massima, ho predisposto nella *repository* una cartella in cui avrei eseguito i rilasci all'ultima versione del prodotto, vista l'assenza di un *package registry* appropriato. Da qui gli sviluppatori della *web app*, hanno potuto poi aggiungere come dipendenza di progetto la mia libreria, aggiornando in base alle evenienze la dipendenza, qualora avessi dovuto correggere dei *bug*.

3.5.1 Verifica della libreria

In parallelo con l'attività di codifica, ho verificato lo sviluppo della libreria sfruttando gli strumenti nativi che erano stati messi a disposizione con *Angular* e di cui avevo fatto precedentemente menzione nella sezione 3.1.

Jasmine è stato lo strumento fondamentale per la realizzazione di tutti i casi di test, dal momento che mi ha semplificato di gran lunga l'attività di verifica permettendomi di implementare due tipi di casi di test.

Test di unità I test di unità mi hanno permesso di verificare le singole unità dell'applicativo, in particolare le classi e i servizi principali che hanno composto il progetto, svolgendo particolare controllo sui metodi e sulle chiamate di ritorno.

Test di integrazione I test di integrazione mi hanno aiutato a verificare il funzionamento delle chiamate nella libreria a un servizio in *API REST*, utilizzando il *server mock* di *Prism*. In previsione di utilizzare un *backend*, ho realizzato i test di integrazione per simulare un'autenticazione con credenziali casuali, verificando che tutte le operazioni previste (chiamata HTTP, salvataggio del *token*, ecc.) venissero invocate correttamente.

Jasmine, inoltre, adotta una sintassi di scrittura molto **semplice** e **verbosa** che mi ha permesso di riconoscere più facilmente i test, visto che utilizza **frasi** per identificare un singolo caso di test.

Questo strumento viene eseguito facendo uso del motore **Karma** che avvia un'istanza del *browser web* (*Google Chrome* di default), eseguendo i test con un ordine casuale e riportando alla fine di tutto anche i risultati di *code coverage* (figura 3.27).



Figura 3.27: Esecuzione di Jasmine nel browser
Fonte: positronx.io

Come esempio, riporto di seguito un estratto di un test di unità, spiegando poi i principali i metodi presenti:

Codice sorgente 3.2: Esempio di implementazione in Typescript di un test con Jasmine

```
describe('SyncSecurity', () => {
  let service: SyncSecurity;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule]
    });
    service = TestBed.inject(SyncSecurity);
  });

  afterEach(() => {
    service.logoutRequest();
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });

  // [...]
  it('should return false if the session is not initialized', () => {
    expect(service.isSessionActive()).toBeFalsy();
  });
  // [...]
});
```

- **describe(...):** permette di descrivere la *test suite* che contiene una serie di test;
- **it(...):** permette di implementare effettivamente il singolo test;
- **expect(...):** permette di verificare il risultato all'interno di un singolo test;
- **beforeEach(...):** esegue delle operazioni prima dell'esecuzione di ogni singolo test;
- **afterEach(...):** esegue delle operazioni dopo l'esecuzione di ogni singolo test;

Buona parte delle operazioni di verifica del *software* le ho automatizzate attraverso l'utilizzo del *server* aziendale per la *Continuous Integration* (sezione 3.1.1), che mi ha notificato a ogni modifica (*commit*) l'eventualità di test falliti.

Per quanto riguarda la **correttezza formale del codice**, ho utilizzato *TSLint* per mantenere le stesse convenzioni nella scrittura con il linguaggio *Typescript*, i cui errori mi venivano segnalati velocemente da *Visual Studio Code*, attraverso un'apposita estensione.

In generale, posso affermare che l'attività di verifica si è rivelata molto **proficua** e mi ha permesso di agevolare lo sviluppo in modo significativo, tanto da produrre un grande quantitativo di test, con un'elevata copertura di casistiche, garantendo così qualità al mio prodotto (figura 3.28).

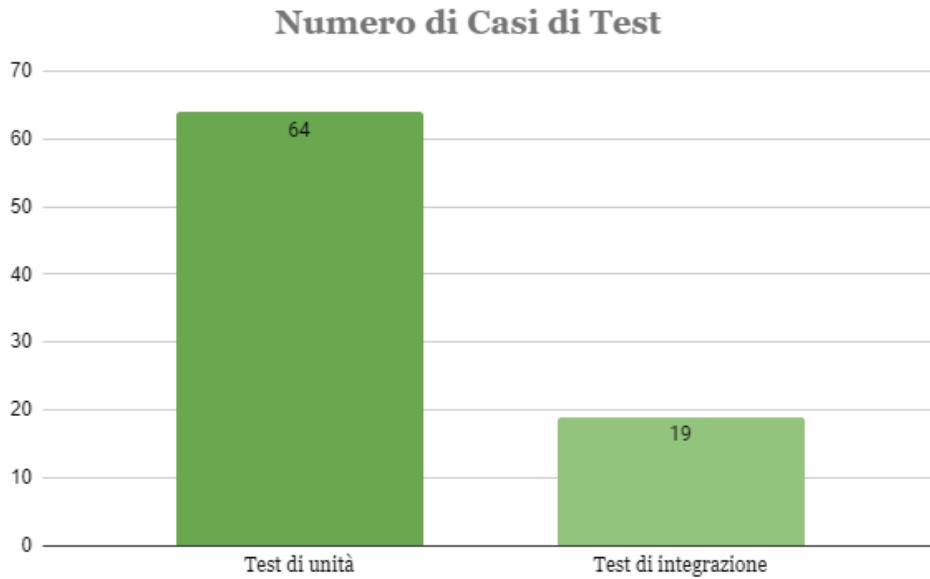


Figura 3.28: Numero di casi di test realizzati nell'attività di verifica

Anche per quanto riguarda il *code coverage* sono riuscito a raggiungere una buona copertura del codice prodotto, rimanendo comunque dentro le tempistiche programmate nel piano di stage (figura 3.29). In linea di massima, l'attività di verifica ha inciso molto sulla correttezza formale e implementativa del *software*, visto che mi ha permesso di correggere un gran numero di possibili *bug*, facilitandomi così nel raggiungere gli obiettivi di qualità del mio prodotto.

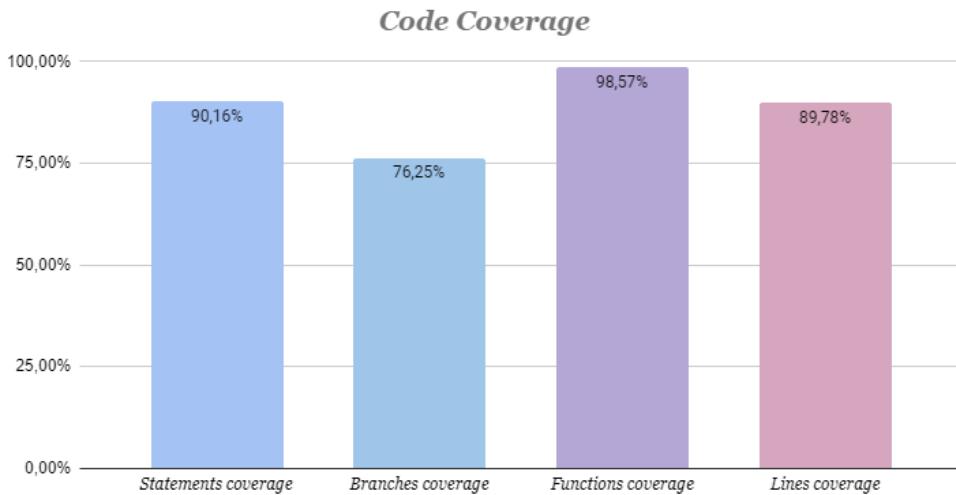


Figura 3.29: Code coverage nell'attività di verifica

3.6 Validazione e collaudo

Nella fase di validazione e collaudo, ho controllato le funzionalità implementate all'interno della libreria a seguito della fase di sviluppo. In questa fase, ho analizzato approfonditamente tutti i requisiti di progetto che avevo precedentemente stilato, valutando per ognuno l'effettiva validità rispetto a quanto atteso da me e dall'azienda. I **risultati** che ho ottenuto sono stati **soddisfacenti** sotto tutti i punti di vista, come è possibile vedere dal seguente grafico (figura 3.30):

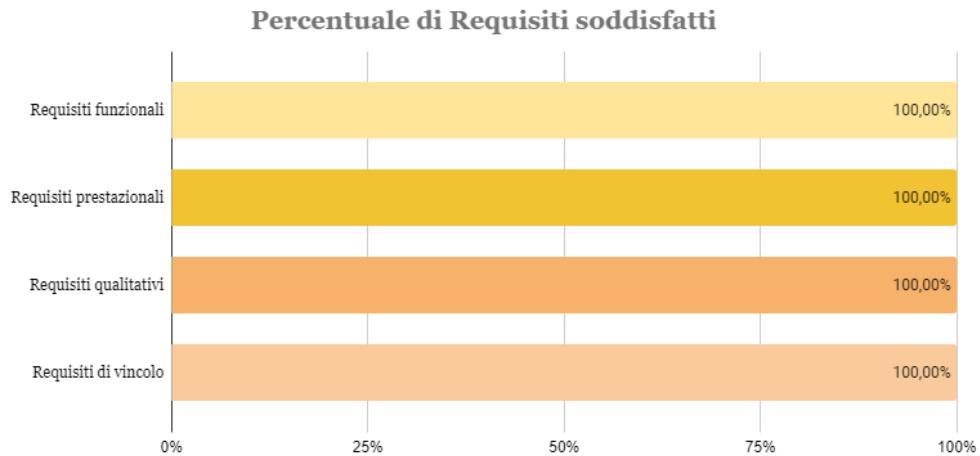


Figura 3.30: Grafico riassuntivo dei requisiti soddisfatti nell'attività di validazione

In questa fase, inoltre, ho riorganizzato i contenuti realizzati in fase di progettazione e ho stilato due documenti che erano stati concordati con il proponente aziendale: il **documento di specifiche tecniche** e il **documento dei casi di test**.

Documento di specifiche tecniche Nel documento di specifiche tecniche, ho riportato per intero tutte le caratteristiche del mio prodotto e ho diviso le singole sezioni come segue:

1. **Introduzione**, in cui ho riportato gli scopi del documento, del prodotto e i riferimenti principali;
2. **Configurazione e installazione**, dove ho inserito i principali requisiti di funzionamento e i singoli comandi per la compilazione, l'installazione e l'avvio della libreria;
3. **Struttura della libreria**, in cui ho illustrato attraverso diagrammi di *package* e diagrammi di classe la struttura generale della libreria, scendendo nel dettaglio sullo scopo e il funzionamento di ogni *package*;
4. **Classi e servizi**, dove ho fornito un'esposizione dettagliata con tanto di firma di ogni singola classe, spiegando per ciascuna lo scopo, le dipendenze, il costruttore, gli attributi e i metodi;

5. **Interazioni della libreria**, dove ho riportato in modo approfondito i diagrammi di sequenza della libreria, segnalando tutti i componenti e le classi coinvolte nel dettaglio;
6. **Casi d'uso della libreria**, come *extra*, in cui ho mostrato i casi d'uso nel dettaglio rifacendomi all'applicazione di esempio che ho creato a titolo dimostrativo e su cui la stessa *web app* del progetto *SyncTrace* si sarebbe comunque dovuta basare.

Documento dei casi di test Nel documento dei casi di test, ho raccolto nello specifico tutti i singoli test realizzati, divisi per classe. A ciascun test ho associato un identificativo, una descrizione e uno stato, che segnalava se il test fosse effettivamente passato o meno. La struttura del documento, pertanto, è stata la seguente:

1. **Introduzione**, in cui ho riportato gli scopi del documento, del prodotto, i riferimenti principali e le convenzioni tipografiche in merito all'identificazione e allo stato dei test;
2. **Configurazione**, dove ho spiegato i requisiti di avvio dei test, i principali strumenti per il *testing*, la scrittura e l'esecuzione di un test;
3. **Struttura e classi**, in cui ho riportato brevemente la struttura della libreria, come nel documento di specifiche;
4. **Test per classe**, dove ho riportato una serie di tabelle per tutte le serie di test realizzati per ogni singola classe, differenziando test di unità dai test di integrazione;
5. **Code Coverage**, in cui ho riportato i risultati di *code coverage* ottenuti fino a quel momento.

Tramite l'utilizzo dell'applicazione di esempio, infine, ho eseguito insieme al propONENTE il **collaudo** della libreria, e ho potuto esporre i punti più importanti del mio lavoro, sottolineando la completezza e l'integrità del mio prodotto.

In aggiunta, i test di integrazione e i test di unità sono stati importanti perché hanno convalidato l'aspetto di sicurezza che l'applicativo web doveva soddisfare, entrando in funzione nel modo in cui era stato pensato dopo le attente scelte progettuali.

3.7 Integrazione del prodotto in SyncTrace

Dopo aver concluso, validato e collaudato la libreria, ho proceduto con integrare il mio prodotto nel resto dell'applicativo *SyncTrace*. Fin dalla fase di progettazione, ho predisposto il mio prodotto con la massima indipendenza affinché offrisse non solo un'integrazione facile, ma anche un buon livello di manutenibilità e di estensibilità. Pertanto, ho mantenuto fin dal principio il mio applicativo come progetto *Angular* a sé stante, in modo che potesse essere aggiunto interamente come modulo del prodotto più grande.

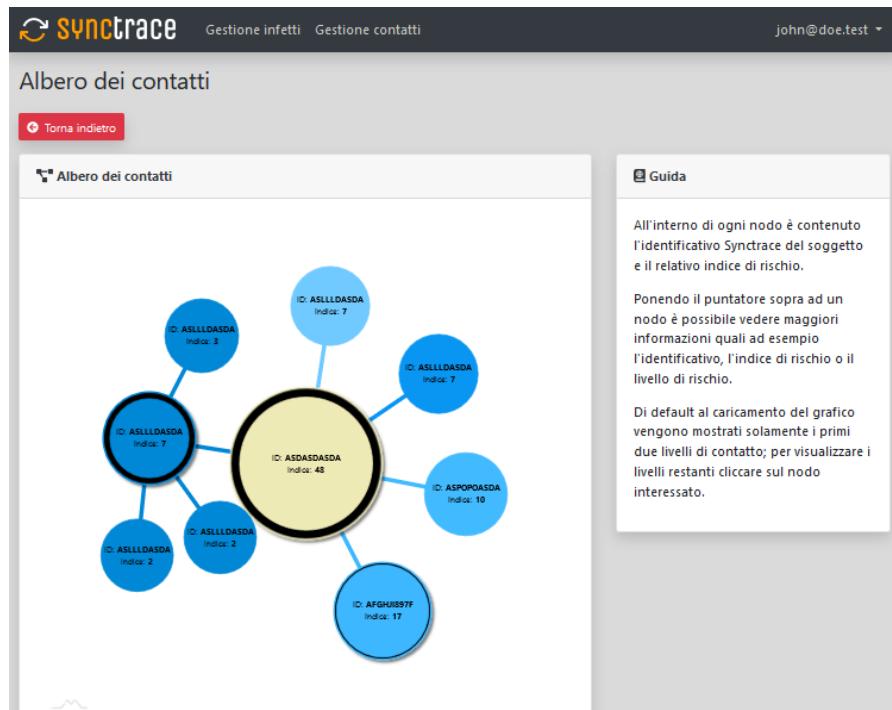


Figura 3.31: Esempio di albero dei contatti dell'applicativo web di SyncTrace

Il progetto *SyncTrace*, come avevo precedentemente illustrato, si compone di 3 componenti principali: l'**applicazione mobile**, il **servizio di backend** in *API REST* e l'**applicazione web**.

- L'**applicazione mobile** permetterà di raccogliere i dati sul *contact tracing*, mettendo in luce i rischi che le persone possono correre quando si entra in contatto con altre persone sconosciute. Da qui, pertanto, ci sarà la sorgente di tutti i dati che verranno rielaborati poi nel *backend*, previo consenso all'invio da parte degli utenti;
- Il **servizio di backend** si metterà in comunicazione con gli altri due componenti, procedendo nell'analisi dei dati e nell'organizzazione delle risorse con opportuni *databases*;
- L'**applicazione web** verrà impiegata per la parte amministrativa, cui gli operatori sanitari potranno accedere per monitorare l'andamento e operare sui dati a loro disposizione.

Fin dal principio del progetto, lo sviluppo di quest'ultimo componente ha avuto diversi obiettivi che hanno portato alla realizzazione di maschere (*view*) per le seguenti sezioni:

- Gestione dei pazienti affetti dal virus;
- Gestione dei contatti;
- Analisi e rappresentazione grafica dei dati pervenuti dall'applicativo mobile (su autorizzazione dell'intestatario dei dati).

I dati gestiti al suo interno, pertanto, sono altamente sensibili e la gestione delle sessioni e della comunicazione tra la *web app* e il *backend* hanno caratterizzato in grossa parte i requisiti (e quindi gli obiettivi) del mio prodotto. Inoltre, non escludo che l'azienda **estenderà** le funzionalità di questo prodotto con l'aggiunta di nuove sezioni (es. sezione per gli esercizi commerciali), dove sarà ancora più importante garantire un **buon grado di sicurezza**, visto il potenziale aumento di utenti.

L'integrazione del mio prodotto ha segnato così un passo molto importante per poter formare un componente essenziale del prodotto finale, la *web app* appunto. Per questa parte, ho collaborato con altri colleghi per spiegare e riportare loro le operazioni che ho messo a disposizione nella mia libreria, cercando di ridurre il più possibile la difficoltà di integrazione dei due prodotti.

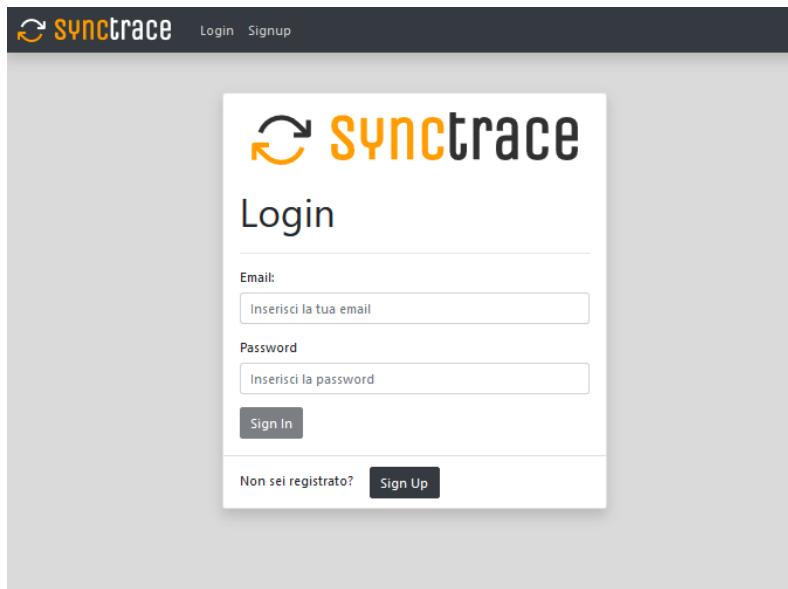


Figura 3.32: Pagina di accesso dell'applicativo web di SyncTrace

Cum grano salis, l'idea di fornire come *package* la libreria si è rivelata efficace, e ha permesso all'azienda di mantenere in modo separato il mio prodotto e di riutilizzarlo anche per altri progetti, senza la necessità di mischiare il codice sorgente nella *repository* di progetto. In tal senso, l'integrazione ha funzionato senza grossi problemi fin da subito e l'*iter* è stato il seguente:

1. Aggiunta del *package* della libreria come dipendenza del progetto per la *web app* sfruttando `npm`;
2. Collegamento e configurazione della libreria nel progetto, per identificarla come *Service*;
3. Breve *refactor* del codice sorgente dei servizi della *web app* per utilizzare le funzioni della mia libreria;
4. Avvio dei test di integrazione della *web app*, per verificare il funzionamento di tutti i componenti;
5. Validazione e collaudo delle funzionalità integrate.

In definitiva, posso affermare che l'ultima operazione ha caratterizzato la **buona riuscita** del mio prodotto e, confrontandomi anche con il proponente, nonché con altri colleghi in azienda, ritengo che ci sia stata piena soddisfazione per tutte le persone coinvolte nel progetto.

Capitolo 4

Resoconto conclusivo

4.1 Raggiungimento degli obiettivi

A conclusione dello stage, ho proceduto insieme al *tutor* aziendale a rivedere gli obiettivi iniziali di stage (sezione 2.7.3) per determinare il loro grado di soddisfazione. In questa analisi, ho valutato sia i prodotti attesi, che gli obiettivi qualitativi e quantitativi del mio prodotto, mettendo in chiaro anche le aspettative che ho soddisfatto e le parti che avrei migliorato in corso d'opera.

Obiettivi di stage Per quanto riguarda gli obiettivi di stage (tabella 2.2), riporto di seguito una tabella riassuntiva con il raggiungimento di ognuno.

ID OBIETTIVO	STATO
0-01	Raggiunto
0-02	Raggiunto
0-03	Raggiunto
0-04	Raggiunto
0-D1	Raggiunto
0-F1	Raggiunto
0-F2	Raggiunto

Tabella 4.1: Tabella riassuntiva degli obiettivi di stage raggiunti

In merito a quanto riportato nella tabella 4.1, ho raggiunto completamente tutti gli obiettivi che mi sono prefissato insieme al proponente aziendale, specialmente quelli facoltativi, che mi hanno aiutato a maturare una migliore conoscenza degli argomenti inizialmente studiati durante l'analisi delle tecnologie.

Obiettivi di prodotto Come chiaramente riportato nella sezione 3.5.1 e nella sezione 3.6, ho condotto l'attività di verifica e l'attività di validazione ottenendo ottimi risultati, che mi hanno permesso di confermare la qualità del prodotto da me realizzato. Per quanto riguarda le soglie di accettazione del *Code Coverage*, inoltre, ho conseguito la copertura minima prevista come riportato nel grafico di seguito (figura 4.1).

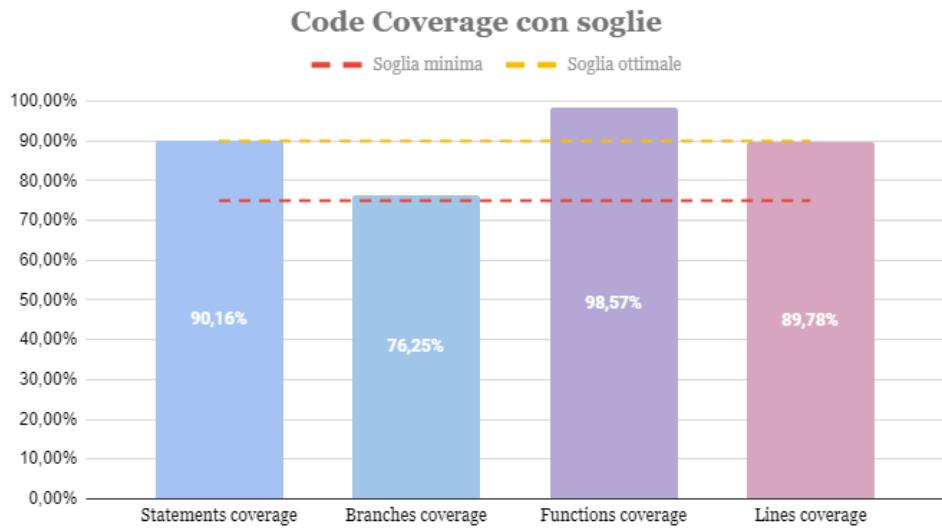


Figura 4.1: Grafico riassuntivo del code coverage raggiunto in base alle soglie

4.1.1 Prodotti realizzati

Per quanto concerne i prodotti attesi dall’azienda (sezione 2.7.3), mi sono impegnato a realizzare tutti i prodotti richiesti, facendo fede alle richieste del *tutor* aziendale.

PRODOTTO	STATO
<i>Proof of concept</i>	Completato e consegnato
Documento tecnico della libreria	Completato e consegnato
Documento di casi di test della libreria	Completato e consegnato
Codice sorgente della libreria	Completato e consegnato

Tabella 4.2: Tabella riassuntiva dei prodotti di stage completati

Ho richiesto il controllo di ciascun prodotto al proponente aziendale, nel corso della realizzazione, in modo da avere un *feedback* continuo sull’andamento del mio lavoro, garantendo così le aspettative primarie dell’azienda. Verso la conclusione dello stage, ho verificato insieme al *tutor* aziendale di aver completato e consegnato tutto ciò che era stato inizialmente pattuito, e il riscontro è stato molto positivo.

Nel codice sorgente, oltre ai componenti della libreria, ho avuto la premura di inserire anche gli *script* sviluppati per il *server* di *Continuous Integration*, descritto precedentemente (sezione 3.1.1).

4.1.2 Aspettative soddisfatte

Nel corso del progetto di stage, posso affermare che gran parte delle aspettative che mi ero posto inizialmente sono state soddisfatte. Grazie a questa esperienza, ho avuto un riscontro positivo sull’azienda sia per quanto concerne il modo di lavorare, che per quanto riguarda l’ambito e i prodotti che sviluppano. La mia ricerca iniziale ha

riguardato principalmente un’azienda che mi potesse offrire un contesto di attualità, con l’utilizzo di tecnologie nuove e innovative, e che mi accogliesse anche in un periodo difficile, ostacolato dalla pandemia.

In *SyncLab*, ho ritrovato questo spirito, merito soprattutto del **tutor aziendale** che mi ha seguito nel corso del progetto e mi ha inserito in un gruppo di lavoro con cui ho potuto confrontarmi costantemente. Lo stesso progetto, inoltre, mi ha permesso di misurare le mie conoscenze e le mie competenze su nuovi ambiti che avevo solo in parte studiato per conto mio e di cui l’università mi aveva dato le basi. Con l’esperienza di progettazione e programmazione individuale, ho potuto creare un *software* per intero che rispecchiasse le specifiche attese, simulando così un’esperienza lavorativa reale. La collaborazione e l’interazione con i colleghi dell’ufficio, inoltre, mi ha permesso di mettere alla prova la mia esperienza di programmatore e di comprendere se le mie idee fossero adeguate o migliorabili nell’ambito del contesto progettuale.

Alcuni punti che **non hanno soddisfatto** le mie aspettative riguardano prevalentemente l’attività analitica iniziale del progetto generale che, a parer mio, è stata in alcune parti carente, anche se comunque sufficiente per me a comprendere gli obiettivi del prodotto che avrei dovuto realizzare. L’assenza di un collega che si dedicasse a pieno sul servizio di *backend* del progetto *SyncTrace*, inoltre, mi ha causato alcuni problemi iniziali, relativi alla realizzazione del *proof of concept* che avrei dovuto presentare in anticipo rispetto alla progettazione e che richiedeva l’uso di un *backend ad hoc*. Ad ogni modo, ritengo comunque che queste mancanze siano del tutto comprensibili visto il periodo di pandemia, e a tal proposito non nego un mio rammarico sul non aver potuto svolgere più tempo in sede aziendale. Il caso migliore che mi ero inizialmente posto includeva la possibilità di partecipare più attivamente in presenza presso la sede di *Sync Lab*, ma questo non è ovviamente imputabile all’azienda che, anzi, non appena possibile, mi ha permesso l’accesso con opportune limitazioni relative alle direttive aziendali interne.

In generale, dunque, ho trovato l’esperienza di stage molto costruttiva sotto tanti punti di vista e, malgrado qualche problema minore, ho potuto comunque fruire di un’ottima offerta di tirocinio, imparando sicuramente moltissimo e chiarendomi molto le idee sul mio percorso futuro.

4.2 Conoscenze professionali maturate

Durante lo stage ho approfondito un gran numero di argomenti che riguardano in particolare i protocolli di sicurezza. Oltre a ciò, l’esperienza di stage mi è stata utile anche per apprendere un metodo lavorativo reale, derivante dal *modus operandi* dell’azienda. Di seguito riporto le principali conoscenze a livello professionale che ho maturato con questo tirocinio.

Approfondimento dei protocolli di sicurezza Nella prima parte dello stage, ho avuto modo di analizzare in modo indipendente una serie di protocolli di sicurezza in ambito web, utilizzati anche a livello *enterprise* da un gran numero di aziende del settore. Nello specifico, ***OAuth 2.0***, per determinare un processo sicuro di gestione delle autorizzazioni all’interno di un applicazione, e ***OpenID Connect***, grazie al quale ho potuto determinare tramite *proof of concept* il suo funzionamento e la sua fattibilità, identificando un caso di utilizzo reale. Di contorno, ho appreso anche il funzionamento del protocollo *HTTPs* e delle politiche *CORS*, nonché del ***JSON Web Token***, utile per lo scambio di informazioni strutturate.

Nuove competenze progettuali Con l’analisi e progettazione di dettaglio ho messo in pratica le mie conoscenze per la realizzazione di classi e interfacce, facendo uso di opportuni ***design pattern*** che mi hanno semplificato l’attività di codifica. Per quanto riguarda i problemi di progettazione, ho portato avanti una serie di soluzioni progettuali, riguardanti la scelta del *Web Storage*, la separazione delle componenti in *package* e l’indipendenza della libreria, che si sono rivelate interessanti per comprenderne l’adeguatezza e l’efficacia a livello implementativo rispetto ad altre soluzioni.

Nuovi linguaggi di programmazione e framework All’inizio dello stage ho studiato e approfondito il ***framework Angular***, nonché tutti gli strumenti ausiliari per l’attività di codifica e verifica (*Jasmine*, *TSLint*). Con l’apprendimento del linguaggio ***TypeScript*** e ***JavaScript***, inoltre, mi sono avvicinato maggiormente allo sviluppo di applicativi *web*, scoprendo nuove funzionalità proposte da questi linguaggi, come le *lambda expression* e gli oggetti *Observables* per gli eventi.

Nuovi strumenti di configurazione del progetto Ho appreso in buona parte il funzionamento di *GitLab*, soprattutto per quanto riguarda le funzionalità relative alla ***Continuous Integration***, mettendo in funzione un *server* nella sede aziendale che rendesse autonomo il mio *workflow*. Ho imparato anche a utilizzare strumenti quali *Stoplight Studio* per la creazione di documenti nello standard *OpenAPI*, ma anche *Trello* per la gestione delle attività, nonché *Google Meet* per le videoconferenze.

Apprendimento del metodo Scrum Nel corso del tirocinio, ho appreso il metodo ***Scrum*** utilizzato dall’azienda e ho imparato principalmente a suddividere e organizzare le mie attività e i miei obiettivi affinché rientrassero nel tempo a mia disposizione. Ho fruito di tutte le risorse a mia disposizione e ho conseguito a pieno anche i principali *meeting*, soprattutto quelli quotidiani, in modalità ***smart working***. Le interazioni avvenute all’interno del contesto lavorativo, inoltre, mi hanno permesso di imparare anche da altri, confrontandomi così nel corso del progetto durante le riunioni orizzontali, e le presentazioni di gruppo mi hanno aiutato a esporre in pubblico i concetti in modo sintetico ed essenziale.

4.3 Valutazione personale

Nel corso del tirocinio, ho maturato diverse idee riguardanti la realtà tra il mondo universitario e il mondo del lavoro. Buona parte delle conoscenze del mondo universitario sono in gran parte essenziali, perché mi hanno aiutato ad avere una *formamentis* e uno spirito critico sul mio operato, ma alcuni degli argomenti trattati mi sono sempre apparsi troppo vecchi rispetto alle necessità del mercato *ICT*. Il mondo del lavoro, comprovato dall’esperienza di stage, mi ha fatto chiarezza su questo aspetto, a tal punto da capire che entrambi i mondi in realtà sono molto complementari tra loro. Ciò che manca al mondo del lavoro, secondo me, è la metodicità e la capacità organizzativa più rigida e ferrea che si ritrova nella vita universitaria, sia dal punto di vista teorico, che dal punto di vista pratico. Tuttavia, mentre nell’attività lavorativa le mansioni e i prodotti sono sempre dinamici e in continua evoluzione, l’università si blocca in un circolo di pensiero antico che, talvolta, può demotivare lo studente o portarlo al disinteresse nei confronti della disciplina. Questo tentativo di avvicinare e coinvolgere gli studenti nel mondo lavoro è sicuramente costruttivo, sia per l’università, la quale dovrebbe imparare più dalle aziende, sia per il mondo del lavoro, che dovrebbe

ascoltare e apprendere dagli studenti i metodi e le idee che potrebbero avvantaggiare un'azienda, senza sfruttarli semplicemente come forza lavoro.

In generale, però, posso affermare che dalla mia esperienza ho avuto solo che piacere di scoprire l'attività lavorativa reale e ringrazio coloro che l'hanno resa possibile. L'aspetto di cui sono stato maggiormente soddisfatto è il progetto, che mi è interessato fin dal principio essendo calato in un contesto di attualità, con un pizzico di innovazione e utilità. L'interazione con i colleghi aziendali, inoltre, mi ha permesso di imparare molto e di conoscere nuove persone con cui potermi confrontare su molti aspetti lavorativi, rimanendo comunque in un ambiente tranquillo e collaborativo. La disponibilità del *tutor* aziendale e del *tutor* interno è stata per me essenziale per avere sempre un riferimento in cui delimitare il mio percorso, senza deviare o rallentare nel corso del progetto.

Per quanto riguarda il percorso universitario, ritengo che in linea di principio ci siano stati contenuti fondamentali, utili per affrontare un tirocinio inserito in molteplici ambiti di mercato. Le lezioni che ho frequentato nel corso della Laurea Triennale mi hanno permesso di imparare gradualmente gli aspetti essenziali di un contesto lavorativo, che sul piano formativo comprendono il metodo di studio, l'organizzazione e la collaborazione nei progetti di gruppo. In aggiunta, le conoscenze teoriche e pratiche di progettazione e codifica, acquisite con i corsi universitari, sono state sicuramente sufficienti nel progetto di stage per poter operare su nuovi linguaggi di programmazione e su *framework* sconosciuti.

La parte più carente che ho ritrovato nell'offerta formativa dell'università riguarda il basso incentivo nell'uso di strumenti di configurazione. In particolare, mi riferisco a strumenti di lavoro molto famosi, come *Git*, che possono essere adottati in modo trasversale per una molteplicità di materie e che non vengono quasi mai menzionati nei corsi in cui si prevedono progetti di gruppo. Questa mancanza, colmata solo da uno studio autonomo o da alcuni corsi dedicati, si fa sentire particolarmente presto non appena si viene a conoscenza della loro esistenza. Se avessi potuto mettere in pratica fin da subito quanto ho appreso nell'ultimo anno di università, avrei sicuramente guadagnato di più in competenze pratiche e, in ottica lavorativa, le stesse competenze sarebbero state molto più apprezzate. Inoltre, una seconda carenza l'ho ritrovata nella poca modernità dei contenuti riportati in alcuni corsi, che in molti casi si sono rivelati meno interessanti, nonostante il loro contributo fosse comunque presente e ben strutturato. In questa circostanza, un aggiornamento degli argomenti trattati in questi corsi potrebbe essere più efficace per stimolare maggior interesse negli studenti, promuovendo così il mondo attuale rispetto al mondo passato, e magari conciliando le principali differenze in un confronto esplicito e motivazionale.

In conclusione, posso confermare di aver ritrovato una grossa utilità nell'esperienza di progetto in vista delle mie intenzioni future, il che mi ha aperto la mente sulla scelta del percorso di Laurea Magistrale da intraprendere. Ho compreso quindi le principali tematiche su cui concentrarmi e le competenze trasversali da maturare, che mi torneranno utili in ottica lavorativa, e ciò mi permetterà di essere più attivo e informato sulle novità e sulle richieste del mercato, così da essere sempre preparato al momento giusto.

Bibliografia

Riferimenti bibliografici

Gamma, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995 (cit. a p. 56).

Siti web

Covid-19 - Italy. URL: <https://covid19.debug.ovh/> (cit. a p. 20).

Don't trust OAuth: Why the "Google Docs" worm was so convincing, ArsTechnica, 2017. URL: <https://arstechnica.com/information-technology/2017/05/dont-trust-oauth-why-the-google-docs-worm-was-so-convincing/> (cit. a p. 45).

End User Authentication with OAuth 2.0. URL: <https://oauth.net/articles/authentication/> (cit. a p. 40).

Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Response Status Codes, RFC 7231. URL: <https://tools.ietf.org/html/rfc7231#section-6> (cit. a p. 49).

Inizio della pandemia e dei focolai in Italia. URL: https://it.wikipedia.org/wiki/Pandemia_di_COVID-19_del_2020_in_Italia#Diffusione_della_pandemia (cit. a p. 14).

JSON Web Token, RFC 7519. URL: <https://tools.ietf.org/html/rfc7519> (cit. a p. 42).

Le misure del governo italiano per l'emergenza COVID-19. URL: <http://www.governo.it/it/coronavirus-misure-del-governo> (cit. a p. 17).

Prima vittima da COVID-19 in Italia. URL: <https://www.ilfattoquotidiano.it/2020/02/22/adriano-trevisan-78-anni-di-vo-euganeo-ecco-chi-e-la-prima-vittima-italiana-del-coronavirus-la-paura-nel-paese-isolato/5713686/> (cit. a p. 14).

S.O.L.I.D: The First 5 Principles of Object Oriented Design. URL: <https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design> (cit. a p. 57).

The OAuth 2.0 Authorization Framework, Authorization Request and Response, RFC 6749. URL: <https://tools.ietf.org/html/rfc6749#section-4.4.1> (cit. a p. 46).

The OAuth 2.0 Authorization Framework, Implicit Flow, RFC 6749. URL: <https://tools.ietf.org/html/rfc6749#section-1.3.2> (cit. a p. 39).

Twitter OAuth API Keys leaked, ThreatPost, 2013. URL: <https://threatpost.com/twitter-oauth-api-keys-leaked-030713/77597/> (cit. a p. 44).