

# Paolino Angeletti

Full Stack Developer

## Informazioni personali

Data di nascita	04/05/1993
Nazionalità	Italiano
Livello inglese	Livello B1

## Contatti

Email	paolinoangeletti@gmail.com
Cellulare	+39 320.29.50.351
Linkedin	linkedin.com/in/paolino-angeletti-64484211b
Indirizzo	via Trivice d'Ossa, 80030, Cimitile, Napoli

## Panoramica professionale

Senior Full Stack Developer con oltre 8 anni di esperienza nello sviluppo di soluzioni enterprise complesse. Specializzato in microservizi backend con Java e Python, sistemi distribuiti, performance e resilienza. Esperienza consolidata in ambito ferroviario, logistica, cloud AWS, test automation e DevOps. Attualmente consulente per Hitachi Rail tramite Teoresi S.p.A., su progetti ad alta affidabilità e criticità.

## Istruzione

### Laurea Triennale in Informatica

Università Federico II, Napoli • 2012–2016 • Votazione: 100/110

### Diploma Perito Commerciale e Programmatore

ITCG Masullo-Theti, Nola (NA) • 2009–2012 • Votazione: 74/100

## Esperienze professionali

### Teoresi Group S.p.A. (Consultant @ Hitachi Rail)

Napoli • Gennaio 2024 – Presente

- Progetti: 360Motion, 360Flow (ambito railway)
- Sviluppo microservizi in Java Spring Boot e Python
- Utilizzo avanzato di: Docker, Kafka, Redis, PostgreSQL, AKHQ, JMeter, SonarQube, TestContainers
- Progettazione e refactoring SDK e librerie interne
- Gestione test di integrazione tramite libreria proprietaria TestCore
- Sicurezza: crittografia, MQTT, autenticazione

## **ADVISE SRL**

Mugnano del Cardinale (AV) • Giugno 2016 – Gennaio 2024

- Progetti Android (Java) & Backend (Node.js, Python)
- Project Manager su vari gestionali (NegoziPerfetto, MBE LAB, MagazzinoPerfetto)
- Integrazione con AWS, RFID, Blockchain, WooCommerce
- Realizzazione APP offline-first sincronizzate con MySQL/DynamoDB
- API REST, notifiche push (Firebase), stampa Bluetooth (ZPL)

## **Software Business SRL**

Tirocinio universitario • Mar 2016 – Ago 2016

- Progettazione e rilascio piattaforma ordini B2B integrata SAP

## **Competenze tecniche**

**Linguaggi:** Java, Python, Kotlin, Node.js, PHP, HTML, CSS, JavaScript, VB.NET, Swift, BeanShell

**Framework / Tooling:** Spring Boot, Flask, Django, Retrofit, Gradle, Maven, Git, GitFlow, Lombok, SonarQube

**Cloud & DevOps:** AWS (Lambda, API Gateway, DynamoDB, Aurora, Cognito, S3, Batch, ECR, EC2, CloudWatch), Docker, Docker Compose, Kubernetes (base), Azure DevOps

### **Database:**

Relazionali: PostgreSQL, MySQL, SQL Server

NoSQL: Redis (anche ReJSON), DynamoDB

**Message Brokers:** Kafka (AKHQ), MQTT

**Testing & CI/CD:** JUnit, JMeter, TestContainers, Swagger, GitHub, Azure DevOps Pipelines

**Pattern usati:** Singleton, Factory, Observer, Mediator

**Documentazione & PM:** Doors, Microsoft Planner, Project Place, Confluence, JIRA

## Approfondimenti tecnici

Le sezioni seguenti includono una descrizione tecnica estesa delle esperienze lavorative, delle tecnologie utilizzate e delle architetture implementate. Sono pensate per chi desidera un approfondimento tecnico sui progetti enterprise seguiti, le soluzioni adottate e il contributo personale.

Esperienze lavorative dettagliate .....	5
Software Business SRL .....	5
ADVISE SRL .....	5
Teoresi Group SPA .....	5
Hitachi Rail STS SPA .....	5
Tecnologie e strumenti .....	7
Linguaggi di programmazione .....	7
Java.....	7
Gradle.....	7
Maven.....	7
Python.....	7
Node.js.....	8
PHP.....	8
HTML, CSS, JavaScript.....	8
VB.NET.....	8
Bean Shell .....	8
Pattern.....	9
Singleton pattern.....	9
Factory pattern.....	9
Observer pattern .....	9
Mediator pattern .....	9
Amazon Web Services - AWS Amazon Cloud .....	10
Aurora DB .....	10
Dynamo DB.....	10
Lambda.....	10
API Gateway .....	10
Cognito .....	10
S3 (Amazon Simple Storage Service) .....	10
Cloud Watch.....	11
Batch.....	11
ECR (Amazon Elastic Container Registry).....	11

Android – sviluppo mobile .....	11
Swift .....	12
Git.....	12
Database .....	12
Database relazionali .....	12
MySQL.....	12
SQL Server.....	13
PostgreSQL .....	13
Database No SQL.....	13
Dynamo DB.....	13
Redis.....	13
Docker .....	14
Messaggistica interna .....	14
MQTT.....	14
Apache Kafka.....	14
ZPL.....	15
RFID.....	15
Documentazione.....	15
Doors .....	15
Gestione attività.....	15
Progetti .....	17
360 Motion .....	17
360 Flow – Veicolo .....	19
TCR - Test Case Runner.....	21
Negozio Perfetto.....	23
MBE LAB .....	25
Magazzino Perfetto.....	25
Quadro Luminoso – QL Mobile.....	27
Apri Un Ticket.....	28
CRM – O .....	29
Movitel .....	30
Acquisto Merci - Creazione Ordini .....	31
Certificazioni .....	32
Corso sicurezza generale .....	32
Corso sicurezza rischio basso.....	33

## Esperienze lavorative dettagliate

### Software Business SRL

Data inizio	Marzo 2016
Data fine	Agosto 2016
Luogo	Mugnano del Cardinale (AV)
Sito web	<a href="http://www.softwarebusiness.it">www.softwarebusiness.it</a>

Questa esperienza è nata come opportunità di tirocinio messa a disposizione dall'università. Il mio obiettivo finale era il rilascio di una piattaforma web user-friendly avente lo scopo di creare ordini clienti in **SAP**. Tale applicativo doveva essere utilizzato dagli stessi customer finali, che quindi potevano ordinare merce autonomamente, stile e-commerce.

### ADVISE SRL

Data inizio	Giugno 2016
Data fine	Gennaio 2024
Luogo	Mugnano del Cardinale (AV)
Sito web	<a href="http://www.advise.it">www.advise.it</a>

Le prime esperienze riguardavano la manutenzione degli applicativi già in produzione, scritti prevalentemente con tecnologia .NET e database SQL o MySQL.

Dopo un anno dalla mia assunzione, sono stati avviati alcuni progetti per lo sviluppo di APP, nello specifico Android e Gestionali di magazzino, sui cui ho lavorato la gran parte del tempo fino ad oggi, e di cui sono diventato Project Manager. I progetti vengono descritti più ampiamente nella sezione dedicata.

### Teoresi Group SPA

Data inizio	Gennaio 2024
Data fine	Esperienza attuale
Luogo	Napoli, Via Ferrante Imparato
Sito web	<a href="http://www.teoresigroup.com">www.teoresigroup.com</a>

Teoresi è una multi-nazionale in ambito di consulenza ingegneristica. Come tale, sono stato assegnato, nel tempo, a diversi progetti per diversi clienti finali.

### Hitachi Rail STS SPA

Data inizio	Gennaio 2024
Data fine	Esperienza attuale
Luogo	Napoli, Via Argine
Sito web	<a href="http://www.hitachirail.com">www.hitachirail.com</a>
Ruolo	Consulente tramite Teoresi S.p.A.

Ho lavorato presso Hitachi come consulente Teoresi.

In particolare, ho lavorato al progetto 360Flow, lato veicolo, per la manutenzione dei moduli già implementati oltre allo sviluppo di nuove librerie e moduli software.

In Giugno 2024, sono stato invece assegnato al progetto 360Motion.

## Tecnologie e strumenti

In questo paragrafo verranno descritte le varie tecnologie e strumenti gestiti negli anni ed il loro contesto di utilizzo.

### Linguaggi di programmazione

#### Java

Ho utilizzato Java nel contesto dello sviluppo Android, implementando funzionalità per applicazioni mobile.

Nel 2024, ho lavorato al progetto 360Motion, implementando servizi distribuiti con SpringBoot all'interno di un'architettura a micro-servizi.

#### Gradle

Ho usato Gradle come strumento di gestione nei seguenti progetti:

- NegozioPerfetto
- MagazzinoPerfetto
- MBE LAB

#### Maven

Ho usato Maven per la gestione di compilazione e dipendenze nei seguenti progetti:

- 360 Motion

#### Python

Ho utilizzato Python in vari progetti ed ambiti:

1. Per la definizione di codice per servizi cloud AWS, sfruttando in particolare la libreria boto3 per l'interazione con i servizi AWS.
2. Nel progetto 360Flow, Python era il linguaggio di programmazione con cui l'intero progetto è stato sviluppato. I moduli sviluppati venivano poi containerizzati in ambito Docker. In questo contesto è stato utilizzato anche un server **Flask** per simulare richieste HTTP.

## Node.js

Ho utilizzato Node.js per la definizione di codice per servizi cloud AWS, sfruttando le sue capacità di gestione degli eventi e delle richieste asincrone.

## PHP

Ho utilizzato PHP per l'implementazione di immagini Docker, facilitando la creazione e la gestione di container per le applicazioni.

## HTML, CSS, JavaScript

Ho utilizzato questa combinazione di linguaggi per la definizione di pagine web, creando interfacce interattive e reattive per gli utenti.

## VB.NET

Nei primi anni della mia esperienza, ho utilizzato VB.NET per l'implementazione di applicazioni web, sfruttando le funzionalità offerte da questo linguaggio.

## Bean Shell

Linguaggio utilizzato per la lettura e modifica di test-case in **JMeter** durante le fasi di bug fixing per il progetto 360 Motion.



## Pattern

Qui di seguito elencherà i vari pattern usati e padroneggiati nel tempo.

### Singleton pattern

Pattern creazione usato per gestire oggetti che dovrebbero essere istanziati una sola volta nel sistema.

Usato in quasi tutti i progetti elencati successivamente.

### Factory pattern

Il Factory pattern è un pattern creazionale che consente di creare oggetti senza specificare esplicitamente la classe concreta di ogni oggetto.

Questo è possibile attraverso l'utilizzo di un'interfaccia o di una classe astratta che definisce un metodo per la creazione di oggetti. Le classi concrete che implementano questa interfaccia decidono quale tipo di oggetto istanziare, in base alle esigenze dell'applicazione.

Questo pattern è ampiamente utilizzato nel progetto 360Flow per gestire la creazione dinamica di vari tipi di oggetti in modo flessibile e scalabile, permettendo una migliore separazione delle responsabilità e facilitando l'estensione del sistema con nuovi tipi di oggetti.

### Observer pattern

Il pattern Observer è un pattern comportamentale che permette a un oggetto, chiamato "soggetto", di notificare automaticamente gli "osservatori" registrati quando avviene un cambiamento nel suo stato. Gli osservatori sono quindi aggiornati e possono reagire di conseguenza.

Questo pattern è stato usato nel progetto 360Flow in modo da mettere in ascolto alcuni componenti dei vari moduli sui subscriber MQTT previsti, in modo da avere un solo componente in ascolto sul broker MQTT e tante classi che sono in ascolto sull'aggiornamento dei valori osservati.

### Mediator pattern

Il pattern Mediator è un pattern comportamentale che favorisce la comunicazione indiretta tra gli oggetti, promuovendo la debole dipendenza tra di essi. Un oggetto "Mediatore" facilita la comunicazione tra gli oggetti "Collegati" attraverso una logica centralizzata, riducendo così il numero di connessioni dirette tra gli stessi.

Questo pattern è stato utilizzato nel progetto 360Flow per facilitare la comunicazione tra componenti diversi del sistema.

## Amazon Web Services - AWS Amazon Cloud

In questi anni ho lavorato quotidianamente con i servizi cloud messi a disposizione da AWS.

Di seguito i servizi usati della piattaforma:

### Aurora DB

Gestione di un database relazione basato su MySQL.

### Dynamo DB

Gestione di un database non relazione serverless basata su struttura key-value.

### Lambda

Servizio di elaborazione che consente di eseguire del codice senza la necessità di effettuare il provisioning o la gestione dei server.

Questo codice storicizzato è poi invocabile tramite API o tramite SDK Amazon.

Il codice per le funzioni di lambda può essere scritto con una lista di linguaggio che la piattaforma mette a disposizione, e nello specifico ho usato: Python, NodeJS.

### API Gateway

Servizio AWS per la creazione, la pubblicazione, la gestione, il monitoraggio e la protezione di API REST, HTTP e WebSocket a qualsiasi livello.

Gli sviluppatori di API possono creare API in grado di accedere ad AWS o ad altri servizi Web, nonché ai dati archiviati nel cloud AWS.

In particolare, nella mia esperienza, ho sempre provveduto a collegare una API ad una funzione Lambda specifica.

### Cognito

Piattaforma di identità per app Web e per dispositivi mobili, utile a gestire l'accesso al sistema tramite credenziali.

È una directory utente, un server di autenticazione e un servizio di autorizzazione per i token di accesso OAuth 2.0 e le credenziali AWS.

### S3 (Amazon Simple Storage Service)

Servizio Amazon dedicato alla archiviazione e gestione di una serie di documenti e file generici previsti dal sistema in implementazione.

Ho gestito nel tempo sia bucket privati che pubblici, in base alla richiesta utente.

### Cloud Watch

Servizio di monitoraggio di AWS per le risorse cloud e le applicazioni in esecuzione su AWS. Ho utilizzato Amazon CloudWatch per raccogliere e tenere traccia dei parametri, raccogliere e monitorare file di log prodotti dai vari servizi gestiti nel tempo.

Esempio: visualizzare i log prodotti da una specifica funzione lambda.

### Batch

Servizio AWS utilizzato per eseguire processi di lunga durata.

Ho utilizzato batch per eseguire, ad esempio, calcoli notturni statistici che in altre condizioni avrebbero impiegato troppo tempo e risorse.

### ECR (Amazon Elastic Container Registry)

Servizio di registro delle immagini container.

Le immagini che ho caricato in ECR erano sviluppate tramite Docker e poi caricate in ECR come repository private.

Tale servizio è stato usato per poter avviare le suddette immagini Docker tramite AWS Batch.

### Android – sviluppo mobile

Il linguaggio Android è stata la tecnologia che ho usato per circa 7 anni, dal 2016 al 2023.

Con Android sono stati implementati la maggior parte dei progetti mostrati successivamente.

Come IDE di sviluppo è stato usato Android Studio, mentre come linguaggio ho usato per lo più Java, ma ho avuto modo di usare anche Kotlin in altri progetti minori e per funzioni specifiche.

Non ho mai usato nessun frame work particolare previsto da Android, mentre ho adottato un'architettura basata su MVC (Model-View-Controller) o MVP (Model-View-Presenter) per garantire una separazione chiara delle responsabilità: le activity fungevano da model o presenter, che invocano fragment che fungono da view, per mostrare l'interfaccia utente e reagire agli input dell'utente. Ho organizzato i file XML per definire il layout dell'interfaccia utente e ho utilizzato i file di risorse per la gestione delle stringhe, delle dimensioni e degli stili.

L'associazione degli elementi della view ed il relativo controllore veniva eseguita tramite view binding.

Inoltre, ho utilizzato Gradle come strumento di build per gestire le dipendenze esterne e assicurare una corretta configurazione del progetto.

Riguardo i database, ho usato sia database locali SQLite sia progetti in cui i dati venivano rilevati tramite API.

Nei progetti **NegozioPerfetto**, **MBE LAB** e **MagazzinoPerfetto**, questo database locale veniva poi sincronizzato con un database in cloud tramite un processo scritto ad-hoc.

Altri servizi usati in questo contesto sono **Firebase**, usato per ricevere notifiche di crash o analisi e per inviare notifiche push, e **Git**, usato per il versioning.

Per quanto riguarda invece l'invocazione di API, si enuncia anche l'utilizzo di **Retrofit**.

## Swift

Nel contesto del progetto **QL Mobile**, ho utilizzato il linguaggio **Swift** per la realizzazione dell'applicazione. La mia conoscenza di questo framework è di base, avendo sviluppato esclusivamente questo progetto utilizzando tale tecnologia. Durante lo sviluppo, ho acquisito familiarità con le principali caratteristiche del linguaggio e le best practice di programmazione **Swift**, consentendomi di creare un'esperienza utente fluida e funzionale nell'applicazione **QL Mobile**.

## Git

Ho utilizzato **Git** come sistema di controllo versione per gestire efficacemente i progetti sviluppati durante la mia esperienza.

Ho utilizzato comandi come **commit**, **push** e **pull** per sincronizzare il mio lavoro con il repository remoto e ho gestito le diverse versioni del codice in modo strutturato utilizzando i **branch**.

In alcuni progetti, ho utilizzato anche **Git Flow**, una alternativa molto valida al **Git branching**.

## Database

Nella mia esperienza professionale, ho lavorato con diversi tipi di database, sia relazionali che non relazionali.

### Database relazionali

Elenco la lista di database relazionali gestiti nella mia esperienza

## MySQL

Database usato nei progetti **NegozioPerfetto**, **MagazzinoPerfetto** e **MBE LAB** come database in cloud per la storicizzazione di dati provenienti dalle APP Android.

Gli strumenti utilizzati sono stati:

- Trigger
- Stored function
- Stored procedure
- Indici

Nel progetto MagazzinoPerfetto, sono stati implementati trigger che invocavano chiamate API che a loro volta invocavano procedure lambda in AWS, per rendere disponibili web hook utili all'utente finale.

Questa soluzione ha consentito l'integrazione fluida con altre applicazioni e sistemi esterni, garantendo una gestione efficiente delle notifiche e degli aggiornamenti in tempo reale all'interno del magazzino.

## SQL Server

Database usato nei progetti CRM-O ed ApriUnTicket, come soluzione usata negli applicativi web menzionati.

## PostgreSQL

Nel 2024, nell'ambito dei progetti Hitachi, ho utilizzato PostgreSQL come database per la storicizzazione dei dati elaborati dal sistema.

## Database No SQL

Elenco la lista di database non relazioni gestiti nella mia esperienza.

### Dynamo DB

Amazon DynamoDB è un servizio di database NoSQL gestito, offerto da AWS, progettato per garantire prestazioni veloci e scalabili. È altamente disponibile, supporta dati strutturati in tabelle, ed è ottimizzato per applicazioni che richiedono basse latenze e alta velocità nelle operazioni di lettura e scrittura.

Nel progetto NegozioPerfetto, è stato usato DynamoDB per la storicizzazione di alcuni tipi di entità, come ad esempio documenti JSON da associare a specifiche transazioni di magazzino memorizzate, invece, in MySQL.

### Redis

Redis è un database in-memory open-source che funziona come un data-store NoSQL. È noto per la sua velocità estremamente elevata, poiché i dati vengono memorizzati principalmente in memoria (RAM) anziché su disco. Redis supporta strutture dati come stringhe, liste, set, hash e bitmaps, ed è spesso utilizzato per caching, session management, real-time analytics e messaging.

Nel 2024, nel progetto 360 Motion, ho usato Redis per la storicizzazione e lettura di JSON che dovevano essere letti e modificati di continuo in runtime.

Usato, inoltre, **RedisCommander** come strumento per l'accesso a tale database.

## Docker

Ho utilizzato Docker come strumento essenziale per la creazione di immagini container, che successivamente sono state caricate su AWS per avviare e gestire processi di medio-lunga durata. Attraverso Docker, ho sfruttato i repository PHP e Python per scrivere le immagini container richieste dagli utenti, garantendo la corretta esecuzione delle applicazioni in un ambiente isolato e riproducibile. Inoltre, ho configurato le immagini Docker per includere tutte le dipendenze necessarie, garantendo una facile distribuzione e scalabilità delle applicazioni.

Nel 2024, nei progetti Hitachi, ho usato Docker e **Docker Compose** per la build dei vari micro-servizi Java implementati.

## Messaggistica interna

Per messaggistica interna intendo la modalità con cui i messaggi vengono distribuiti e gestiti all'interno di un'architettura basata su micro-servizi. In un contesto di micro-servizi, le applicazioni sono suddivise in componenti più piccoli e autonomi che comunicano tra loro attraverso scambi di messaggi. Questi messaggi possono includere richieste di servizio, notifiche di eventi o dati aggiornati. La messaggistica interna è cruciale per garantire la coerenza, l'affidabilità e la scalabilità dei servizi distribuiti, facilitando l'integrazione tra i vari componenti e assicurando che le informazioni fluiscano in modo efficiente attraverso l'intero sistema.

## MQTT

MQTT è un protocollo di messaggistica leggero e basato su publish/subscribe, progettato per reti con larghezza di banda limitata e connessioni intermittenti.

Nel 2024 ho usato questo protocollo nei progetti 360 Flow Veicolo e 360 Motion.

In particolare, questo protocollo veniva usato dai veicoli in movimento per inviare ai componenti centralizzati dati di vehicle-congestion, vehicle-position e simili.

## Apache Kafka

Apache Kafka è una piattaforma di streaming distribuito e altamente scalabile, progettata per gestire flussi di dati in tempo reale. Basato su un'architettura a log distribuiti, Kafka consente di pubblicare, sottoscrivere e archiviare flussi di record in modo resiliente e scalabile. È particolarmente utile per gestire grandi volumi di dati in movimento e per costruire pipeline di dati e applicazioni di analisi in tempo reale.

Ho usato questo strumento nel 2024 nel progetto 360Motion, per scambiare messaggi tra i vari componenti del sistema.

Usato **AKHQ** per l'accesso a Kafka per la lettura dei topic e dei vari messaggi in circolazione.

## ZPL

Ho utilizzato lo ZPL per creare etichette stampabili tramite stampanti Zebra, consentendomi di gestire le dimensioni delle etichette e altri aspetti legati alla funzionalità. Questo linguaggio si è dimostrato particolarmente utile per le stampe via Bluetooth, riducendo il carico di dati da inviare dal dispositivo alla stampante, ottimizzando così il processo di stampa.

## RFID

Nel contesto del progetto MagazzinoPerfetto, ho applicato la tecnologia RFID per facilitare il tracciamento dell'entrata e dell'uscita della merce, nonché per semplificare le attività di inventario e il controllo dello stock in magazzino. In particolare, abbiamo utilizzato l'area di memoria EPC per memorizzare i dati univoci degli articoli, mentre il TID è stato sfruttato per monitorare il numero di rilevazioni dell'EPC specifico durante la lettura RFID.

## Documentazione

Di seguito vari strumenti usati per documentare i software a cui ho lavorato.

## Doors

IBM Engineering Requirements Management DOORS è uno strumento di gestione dei requisiti ampiamente utilizzato nel settore dell'ingegneria e dello sviluppo di software.

Ho utilizzato questo strumento nel 2024 per gestire i requisiti del progetto 360 Motion.

## Gestione attività

Gli strumenti utilizzati per la gestione di attività interne sono stati:

- Planner

Microsoft Planner è uno strumento di gestione dei progetti e delle attività integrato in Microsoft 365.

- Project Place

Project Place è una piattaforma di gestione dei progetti e delle collaborazioni online, sviluppata da Planview.



# Progetti

## 360 Motion

Azienda  
Sito web

Hitachi Rail - Teoresi S.p.A.  
[www.hitachirail.com/smart-mobility/360motion](http://www.hitachirail.com/smart-mobility/360motion)

360Motion è una piattaforma pionieristica che sta trasformando la mobilità urbana consolidando i dati provenienti da diversi servizi di trasporto in una soluzione unificata. È alimentato da algoritmi avanzati di intelligenza artificiale e ML, che gli consentono di ottimizzare dinamicamente le reti di trasporto pubblico, facilitare la pianificazione degli scenari e anticipare e mitigare le interruzioni.

In pratica, 360Motion riunisce i dati di più servizi di trasporto (bus, metro e veicoli vari) in un'unica piattaforma unificata.

Questi dati vengono ricevuti ed analizzati, con lo scopo di fornire una piattaforma web che possa fornire all'utente quante più informazioni e operatività possibili.

360Motion è una piattaforma indipendente dall'hardware, in grado di scalare le soluzioni di mobilità esistenti raccogliendo e acquisendo dati da qualsiasi tipo di fonte e consentendo a operatori e passeggeri di comprendere il modo ottimale di spostarsi attraverso i paesaggi urbani.

Tecnicamente, invece, 360Motion è un progetto a micro-servizi, scritti in Java (SpringBoot), Python, ed una parte front-end scritta in Angular, che viene distribuito tramite Kubernetes.

In questa mia esperienza come consulente, le tecnologie utilizzate sono state:

- Java con SpringBoot, IntelliJ Idea come IDE
- Maven come strumento di gestione
- PostgreSQL, con PgAdmin come strumento di accesso e gestione.
- Docker
- Swagger
- Git, come strumento di versioning
- Kafka, come distributore interno di messaggi, ed AKHQ come strumento di lettura dei vari topic.
- Redis, ossia un database NoSQL in cache per ottimizzarne le letture. Usato RedisCommander come strumento per accedervi.
- Azure DevOps, come strumento di accesso alle repository e esecutore di pull-request su main, pipeline per compilazione e rilascio automatico.
- SonarQube, strumento per eseguire analisi statica del codice
- JMeter per l'attività di verifica e fix dei work-item rilevati, con linguaggio BeanShell
- Doors, strumento usato come repository dei requisiti di sistema, ove eventualmente applicare modifiche ad essi.
- Project Place, strumento per la gestione dei task interni.
- Cisco Any Connect, usato per accesso a VPN interne.

Invece, le attività di cui mi sono occupato sono:

- Implementazione di Integration-test per controller Spring che ne erano sprovvisti, con JUnit.
- Incremento della code coverage del codice.
- Integrazione di Swagger nei controller previsti.
- Verifica ed eventuale fix per work-item rilevati dal team di V&V (Verification & Validation)
- Refactoring di un SDK interno utile alla gestione delle proprietà che i singoli componenti potevano richiedere al database
- Implementazione della libreria **TestCore**, utilizzata come framework per la scrittura di integration test.  
Questa libreria mette a disposizione vari connettori (database, Redis, Kafka, ecc.) e gestisce, tramite il file **application.properties**, l'accesso a tali connettori tramite test container o tramite l'ambiente reale locale.  
I connettori vengono implementati utilizzando il **factory pattern**, garantendo così una facile aggiunta di nuovi elementi.
- Implementazione di un nuovo microservizio Java, **platform-integration-test**, che, tramite la libreria TestCore, contiene una serie di test di integrazione per verificare il comportamento dei vari microservizi rilasciati.

Nel contesto di IntelliJ, sono stati usati i seguenti plugins:

- Lombok, usato per importare annotazioni e ridurre il codice scritto
- SonarLint, utile a rilevare eventuali problemi del codice

## 360 Flow – Veicolo

Azienda  
Sito web

Hitachi Rail - Teoresi S.p.A.  
[www.hitachirail.com](http://www.hitachirail.com)

360Flow è un sistema a micro servizi per il conteggio delle persone a bordo veicolo.

Il sistema è stato sviluppato in Python su più moduli che comunicano tra di loro tramite algoritmo MQTT e tramite l'esterno tramite HTTP e socket.

Il sistema si interfaccia con la sensoristica per la manipolazione dei dati di congestionamento a bordo treno, dei dati di posizionamento e dei dati di diagnostica.

I moduli sono sviluppati per container Docker su sistema operativo Windows. Il sistema utilizza dei sistemi per difendersi da attacchi informatici. Inoltre è presente la possibilità di interfacciamento diretto con il TCMS a bordo treno tramite un modulo sviluppato in C con protocollo TRDP.

Inoltre è presente un modulo di AI per la rilevazione delle porte aperte o chiuse in caso di impossibilità di connessione con il TCMS. È stato sviluppato un proxy per consentire la connessione da parte di app terze, come ad esempio HMI per di manutenzione che permette l'invio di comandi al sistema o la lettura di dati da esso. È presente anche un database di tipo relazionale PostgreSQL il quale viene utilizzato per la configurazione dei dispositivi e per la storicizzazione dei dati. Si è dovuta gestire tutta la parte relativa alla sicurezza informatica per le comunicazioni in ingresso ed in uscita dal sistema, in particolare utilizzando certificazioni sulle comunicazioni, crittazione dei dati con chiavi simmetriche e asimmetriche e combinazioni utente password per l'autenticazione.

Tra i dati che la 360Flow raccoglie possiamo nominare:

- Numero di persone entrate ed uscite ad una certa stazione per una certa porta del treno
- Numero di persone che, durante il viaggio, ha cambiato la propria carrozza di viaggio.
- Tipo di oggetti che occupano il treno, esempio biciclette, carrozzine, ecc.

360Flow prevede software sia per la parte veicolo (treno) che per la parte di stazione.

In particolare, ho lavorato alla parte di veicolo, eseguendo i seguenti sviluppi:

- Implementazione delle operazioni CRUD sul database PostgreSQL.
- Gestione del formato JSON dei messaggi che il modulo ActionCore (proxy) doveva prevedere.
- Gestione dell'algoritmo di crittazione che doveva essere applicato ai messaggi da pubblicare tramite MQTT.
- Sviluppo della libreria centralizzata per il logging del sistema.
- Sviluppo di un componente del modulo di diagnostica, il FileArchiver, utile a gestire i log creati nel punto precedente. Di fatti, il FileArchiver verifica se i log abbiano superato una certa dimensione oppure una certa data di creazione, ed in tal caso vengono archiviati ed inviati al sistema esterno centrale.
- Sviluppo del modulo DiagnosticReporter, modulo dedicato al monitoraggio di alcuni valori del sistema centrale.

Esempio, se la RAM della macchina a bordo treno supera alcuni limiti previsti, il sistema di controllo centrale deve essere allertato tramite messaggi HTTP e/o MQTT.

Le **tecnologie** e strumenti utilizzati sono stati:

- **Python** come linguaggio di programmazione dei moduli, e **PyCharm** come IDE di sviluppo.
- **Docker** per la creazione di immagini per ogni modulo previsto, così da rendere il funzionamento di un container (modulo) indipendente dal funzionamento degli altri.
- **CodeCommit + GIT** per il versioning del progetto.  
Il team di sviluppo era composto da 7 persone lato veicolo, che potevano interagire anche sugli stessi file.  
In particolare, è stato usato **Git Flow** per assolvere alle funzionalità di versioning.
- **MQTT** per lo scambio di messaggi tra i moduli.  
Essendo questo un software da installare su macchina a bordo treno, la connessione poteva essere instabile, ecco perché si è scelto di usare questo protocollo anziché l'HTTP per lo scambio di messaggi.
- **PostgreSQL** come database per i dati raccolti. È stato selezionato PostgreSQL come database a causa della sua capacità di gestire un alto volume di scritture simultanee, necessario per il sistema.

## TCR - Test Case Runner

Azienda | Progetto interno presso Teoresi S.p.A.

Il Test Case Runner è una piattaforma generica per implementare l'automazione dei test, basata su un'applicazione web **Python Django**.

Le funzionalità principali consentono di:

- caricare una suite di test basata su un formato **XML** validato
- caricare plugins su richiesta
- memorizzare su un database **SQLite** le informazioni del test (ovvero test, descrizione, requisiti, report di esecuzione)
- eseguire le fasi di prova
- esportare documenti personalizzati in base alle informazioni archiviate nel database.

L'applicazione può essere personalizzata per testare un progetto specifico.

Il core carica automaticamente i plugins presenti in una cartella specifica.

I plugin possono implementare le seguenti interfacce:

- pre-esecuzione: una serie di passaggi eseguiti prima dell'esecuzione dei passaggi di test (ovvero preconfigurazione del test)
- eseguire: una serie di passaggi di test
- post-esecuzione: una serie di passaggi eseguiti dopo l'esecuzione dei passaggi di test (ovvero smontaggio)

L'applicazione web Python Django consente di:

- caricare una suite di test
- creare una nuova suite di test
- modificare una suite di test (ad esempio aggiornando titolo o descrizione, aggiungendo test case, modificando passaggi di test, ...)
- eseguire una suite di test
- mostra la cronologia della suite di test
- Esportare una suite di test.

In questo progetto mi sono occupato di:

- Migrazione del progetto da **Python 2.7** a **Python 3.9**, in cui sono stato coinvolto nel refactoring di alcune funzionalità, dato che alcune librerie usate in Python 2.7 non erano presenti per Python 3.9 oppure producevano risultati diversi. Nomino ad esempio la libreria `py_spy`, sostituita con `py_shark` per lo sniffing di pacchetti di rete UDP.
- Integrazione del protocollo **TRDP**, oltre a quello già presente **IPTCOM**.  
Il TRDP (Train Real-time Data Protocol) è un protocollo di comunicazione progettato

specificamente per il settore ferroviario, che permette la trasmissione di dati in tempo reale tra i vari sistemi di un treno. È stato sviluppato per soddisfare le esigenze di comunicazione affidabile e deterministica in ambienti ferroviari, dove è fondamentale avere un basso ritardo e alta disponibilità dei dati.

- Definizione di una **GUI** utile alla creazione dei test XML in maniera rapida ed intuitiva.

Le **tecnologie** e strumenti utilizzati sono stati:

- Git (con **Bitbucket**) come sistema di versioning.
- **JIRA** per la gestione del progetto con Agile Scrum.
- **Confluence** per la gestione della documentazione.
- **Python Django** for the Web application.

## Negozio Perfetto

Azienda	Advise SRL
Sito web	<a href="http://www.negozioperfetto.it/">http://www.negozioperfetto.it/</a>
Google Play Store	<a href="#">NegozioPerfetto APP</a>

NegozioPerfetto è un progetto nato dall'idea di aiutare i piccoli negozianti a contrastare le grandi multinazionali che oggi dominano il mercato.

NegozioPerfetto è quindi un gestionale light, sviluppato per essere quanto più intuitivo e semplice possibile.

Oltre alle classiche funzioni di gestione della movimentazione di magazzino (quindi carico, scarico, trasferimenti, ecc.), l'obiettivo del progetto era quello di implementare un sito e-commerce completamente ed automaticamente integrato al gestionale, così da mostrare online sempre anagrafiche e stock aggiornati in tempo reale.

Altra questione affrontata nel progetto era l'utilizzo della APP in offline: dato che, in un negozio, la connessione potrebbe essere scarsa o comunque non perfetta, il negoziante avrebbe potuto tranquillamente vendere.

Tramite poi un processo di sincronizzazione, i dati venivano sincronizzati sia in entrata che in uscita con il cloud, permettendo quindi sia un backup dei dati sia la possibilità di lavorare offline con una lista indefinita di dispositivi.

In questo progetto sono stato coinvolto nelle seguenti fasi del software:

- Analisi
- Design
- Implementazione
- Manutenzione.

Di seguito, invece, le tecnologie usate:

### Amazon AWS Cloud

Come detto nella descrizione del progetto, la APP prevede di lavorare in offline per poi sincronizzare le modifiche (sia in importazione che esportazione) su un server in cloud. È stato scelto AWS come piattaforma cloud, di cui sono stati utilizzati svariati servizi:

- **EC2**: servizio per avere in esecuzione un server Linux.
- **Aurora DB**: database relazione MySQL.
- **DynamoDB**: database non relazione usato per contenere alcune informazioni particolari.
- **Lambda**: servizio con cui creare delle procedure online raggiungibili tramite API. Per la scrittura di queste lambda, sono stati usati principalmente due linguaggi: **Python** e **NodeJS**.
- **API Gateway** : servizio con cui creare delle API associate poi a delle funzioni lambda.
- **Cognito** : servizio con cui implementare velocemente le procedure di login, recupero password e registrazione al sistema.

- **S3** : spazio di storage in cui poter salvare file.
- **Iot Core** : servizio di IOT usati per richieste verticali, esempio implementazione di un Dash button con cui ordinare uno specifico articolo solamente cliccando il tasto previsto.

## **Java (Android)**

La APP è stata scritta utilizzando il linguaggio Android nativo, mentre come IDE è stato usato Android Studio.

È stato implementato un database in locale SQLite che poi, tramite API, veniva sincronizzato con il database in cloud in MySQL.

Altri strumenti usati durante il progetto sono stati Fabric per l'Analytics e Crashlitycs e Firebase per l'invio di notifiche push al sistema.

## **Python, NodeJS**

Linguaggi utilizzati per scrivere le funzioni lambda necessarie al sistema.

## **Git Hub**

È stato usato GitHub come piattaforma di versioning.

## **WooCommerce**

Il sito integrato al progetto è stato creato tramite piattaforma WooCommerce, ed ovviamente sono state implementate tutte le API utili al progetto per implementare l'integrazione con il sito discussa nella descrizione del progetto.

## **MySQL**

Come database in cloud è stato scelto una soluzione basata sul MySQL.

Negli anni sono stati approfonditi varie tematiche relativamente al database, ossia sono state implementati trigger, stored functions, stored procedures ed indicizzazione delle tabelle più sollecitate dalle operazioni previste.

## **ZPL**

Linguaggio utilizzato per creare etichette stampabili tramite stampanti Zebra, in modo da gestire dimensione etichette ed altri aspetti connessi alla funzionalità.

Particolarmente utile per stampe Bluetooth, in modo da alleggerire il carico da inviare da dispositivo a stampante.



## MBE LAB

Azienda	Advise SRL
Sito web	<a href="http://www.mbe.it">www.mbe.it</a>
Google Play Store	<a href="#">MBE LAB APP</a>

MBE LAB è un gestionale per logistiche nato dall'interesse di Mail Boxes Etc. verso NegozioPerfetto.

Essendo però NegozioPerfetto pensato per negozianti e più concentrato sulla vendita al banco, ecco che viene avviato questo nuovo progetto di logistica.

Se da un lato l'infrastruttura non ha subito grosse modifiche rispetto a NegozioPerfetto (incremento del numero delle tabelle, campi diversi), la APP è stata completamente rivisitata, dato che i requisiti tra la vendita al banco e la logistica sono parecchio differenti.

Anche sotto l'aspetto delle tecnologie il sistema non ha subito grosse modifiche.

Dunque, le differenze tra i due progetti sono per lo più funzionali che infrastrutturali.

In questo progetto ci siamo scontrati con l'enorme mole di dati che logisticamente venivano prodotti, e ciò ha comportato alcune migliorie al database ed alla APP stessa in termini di performance.

Basti pensare che, in alcuni casi particolare, la APP ha dovuto gestire, sincronizzando automaticamente più di 50 terminali, le emissioni di circa 10.000 documenti di spedizione al giorno.

Le funzionalità si sono poi concentrate sull'integrazione tra i sistemi da noi gestiti e quelli di MBE, implementando una lista corposa di API (lettera vettura, spedizioni, utenza) ed un sistema di SSO (single sign-on).

Ad oggi MBE LAB è un progetto utilizzato in vari paesi del mondo, quali Italia, Germania, Inghilterra, Spagna, Polonia, Kazakistan, Panama e Francia, e la lista con il tempo potrà solamente aumentare.

In questo progetto sono stato coinvolto nelle seguenti fasi del software: Analisi, Design, Implementazione e manutenzione.

## Magazzino Perfetto

Azienda	Advise SRL
Sito web	<a href="http://www.magazzinoperfetto.it">www.magazzinoperfetto.it</a>
Google Play Store	<a href="#">MagazzinoPerfetto APP</a>

Dopo l'esperienza fatta con MBE LAB, dunque, si arriva al lancio di un nuovo progetto: MagazzinoPerfetto.

MagazzinoPerfetto nasce anch'esso come gestionale, ma questa volta non di negozi, non di logistiche, ma magazzini nel loro completo ciclo di vita.

Quindi, anche in questo caso, le modifiche di infrastruttura sono ben poche (incremento del numero delle tabelle, campi diversi), ma sono maggiormente modifiche funzionali ed integrazioni verticali verso altri sistemi.

Infatti, avendo dovuto implementare un WMS completo, ci siamo ritrovati a doverci interfacciare

con molteplici sistemi di terzi parti, in base alle esigenze dei clienti.

Tra le principali integrazioni eseguite:

- Integrazione con un sistema di blockchain, con cui memorizzare le transazioni della merce.
- Implementazione della tecnologia RFID.
- Integrazione con sistemi di contabilità.
- Integrazione con la dogana francese.
- Integrazione totale con Zebra, azienda di hardware di magazzino (terminali, RFID, visori).
- Integrazione con WooCommerce per siti customizzati.

In questo progetto sono stato coinvolto nelle seguenti fasi del software: Analisi, Design, Implementazione e manutenzione.

## Quadro Luminoso – QL Mobile

Azienda | Ferrovie dello Stato Italiane

Progetto commissionato da Trenitalia, che aveva l'esigenza di monitorare, in maniera sicura e semplice, le varie tratte adibite all'utilizzo del sistema.

Tutt'ora il sistema è in uso dalle stazioni di Bologna e Milano, ma col tempo la lista potrà solamente aumentare.

QL Mobile è una APP che, tramite sistemi di API, comunica con i sistemi RFI per ottenere informazioni riguardo le tratte, i ritardi, le videoregistrazioni, che un qualsiasi operatore poteva interrogare per qualsiasi esigenza.

Oltre ciò, la APP implementa anche un sistema di streaming video per accedere a varie telecamere poste su una specifica tratta.

Per ragioni di sicurezza, la APP non è pubblicata sul PlayStore ufficiale ed opera in maniera corretta solo in presenza di Rete mobile RFI sul device, così da controllare con facilità i dispositivi da cui la APP può funzionare.

La APP è stata rilasciata sia per Android che per iOS, scritta in entrambi i casi con linguaggio nativo Android e Swift.

Strumenti coinvolti:

- Android (Java).
- Swift (IOS).
- API.

## Apri Un Ticket

Azienda

| Advise SRL

ApriUnTicket è un CRM (Customer Relationship Management) sviluppato dapprima per uso interno aziendale, e poi venduto a clienti terzi per alcune funzionalità utili individuate in esso. Questo CRM, a dispetto degli altri, è un CRM sviluppato con l'obiettivo di facilitare non solo il rapporto con la clientela e la gestione dei vari problemi informatici che ogni giorno possono venire a mostrarsi, ma è stato sviluppato con il vero scopo di aiutare i tecnici che lavorano ai problemi, e quindi vien data la possibilità di indicare metodo (Teleassistenza oppure On Side) e durata di un intervento, così da poter poi estrarre anche reportistiche utili a capire, ad esempio, il cliente più impegnativo, il cliente con meno problemi, e così via.

Altra particolarità di questo progetto (presente però anche in altri CRM) è l'implementazione di un lettore automatico di email, cioè un utente con problema non dovrà far altro che scrivere una email ad un indirizzo specificato, e tale lettore automatico aprirà un ticket automaticamente, oppure lo aggiornerà se la email richiama un ticket già aperto in precedenza.

In questo modo sia gli utenti che i tecnici non dovranno mai accedere al portale web del CRM per gestire un issue (o ticket), ma potranno farlo in maniera comodissima tramite le email.

L'applicativo è stato scritto tramite tecnologie .NET (ASPX nello specifico) con database SQL per la gestione dei dati.

Per la parte di front-end invece sono stati utilizzati:

HTML, CSS, Javascript.

IDE di sviluppo: Visual studio.

PHP: usato per l'implementazione del lettore automatico di email.

## CRM – O

Azienda

| Advise SRL

CRM – O (Officina) è un software sviluppato per una delle più grandi concessionarie Peugeot e Volkswagen della Campania.

Il software nasce come esigenza alla seguente richiesta: le due grandi multinazionali nominate sopra, richiedono con cadenza mensile un file tracciato, in formato txt, contenente l'elenco dei clienti soddisfatti e non passati in officina durante il mese corrente.

La soddisfazione viene stabilita tramite un colloquio telefonico che viene effettuato.

Dunque, il software, aveva i seguenti compiti:

- Comunicare con i gestionali, ottenendo così i clienti da chiamare questo mese.
- Per rendere agile la chiamata, ossia senza l'onere di dover digitare il numero da chiamare, tale software comunicava con un sistema di chiamate **VoIP** ([www.voispeed.com](http://www.voispeed.com)), in modo che l'operatore potesse effettuare chiamate direttamente dalla pagina web.
- Compilazione dei questionari, da effettuare, ovviamente, a chiamata conclusa col cliente (o durante).
- Generazione del file tracciato, rispettando le regole previste, ed invio di tale file alle case madri.

Applicativo scritto in .NET, e database SQL per l'archivio dei dati.

Questo piccolo progetto nasce dalla completa integrazione con i sistemi Voispeed, azienda fornitrice di centralini professionali aziendali.

Questa applicazione aveva l'obiettivo di interfacciarsi con tale centralino in cloud direttamente da un applicativo desktop che implementava le funzioni richieste dall'analisi.

### **Tecnologie**

- Microsoft .NET (VB.NET) come linguaggio di programmazione.
- HTML, CSS, Javascript per la parte front-end.
- SQL come database.
- API per comunicare con il centralino.

## Acquisto Merci - Creazione Ordini

Azienda | Software Business SRL

Questa applicazione è stata implementata durante il mio tirocinio aziendale.

Essa nasce dall'esigenza dei clienti finali di ordinare merce in autonomia, e quindi l'applicativo, implementato in stile Master-detail, mostrava la lista di articoli acquistabili e la gestione di un carrello finale.

Dopodiché, ad ordine completo, esso veniva esportato nel sistema SAP di riferimento.

### **Tecnologie**

- HTML, CSS, Javascript per la parte front-end.
- XML e JSON per lo scambio di dati.
- API per creare l'ordine nel sistema cliente previsto.
- SAP WEB IDE come strumento di programmazione.

## Certificazioni

### Corso sicurezza generale



#### ATTESTATO DI FREQUENZA E PROFITTO

### Corso di Formazione Generale alla Salute e Sicurezza per i Lavoratori

Il corso di formazione è stato svolto in ottemperanza al D. Lgs. 81/08, art. 37, comma 7 e s.m.i e all'Accordo Stato Regioni del 21/12/2011

conferito a:

**Angeletti Paolino**

codice fiscale

**NGLPLN93E04G812F**

Ore di formazione previste	Ore di formazione frequentate
<b>4</b>	<b>4</b>

Principali argomenti trattati nel corso dell'azione formativa:

1. Concetti di rischio;
2. Danno;
3. Prevenzione;
4. Protezione;
5. Organizzazione della prevenzione aziendale
6. Diritti, doveri e sanzioni per i vari soggetti aziendali;
7. Organi di vigilanza, controllo e assistenza.

Soggetto erogatore: **Teoresi S.p.A.**

con sede a **Napoli** in **Via Ferrante Imparato, 198**

La formazione si è svolta dal **27/03/2024** al **27/03/2024**

Data **04/04/2024**

Il Datore di Lavoro  
(timbro e firma)

  
**Teoresi S.p.A.**  
via Perugia, 24 - 10152 Torino  
c.f. e p.iva 03037960014

Il docente

Costabile Lo Schiavo





## ATTESTATO DI FREQUENZA E PROFITTO

### Corso di Formazione Specifica alla Salute e Sicurezza per i Lavoratori RISCHIO BASSO

Il corso di formazione è stato svolto in ottemperanza al D. Lgs. 81/08, art. 37, comma 7 e s.m.i e all'Accordo  
Stato Regioni del 21/12/2011

conferito a:

**Angeletti Paolino**

codice fiscale

**NGLPLN93E04G812F**

Ore di formazione previste	Ore di formazione frequentate
<b>4</b>	<b>4</b>

Settore Ateco 2007: 62.02.00

Principali argomenti trattati nel corso dell'azione formativa:

1. Rischi infortuni	2. Stress lavoro correlato
3. Meccanici generali	4. Movimentazione manuale dei carichi
5. Elettrici generali	6. Segnaletica
7. Emergenze	8. Rischi specifici associati alla mansione
9. Organizzazione del lavoro	10. Procedure esodo e incendi
11. Ambienti di lavoro	12. Tutela della lavoratrice in gravidanza e allattamento

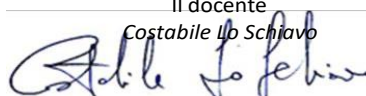
Soggetto erogatore: **Teoresi S.p.A**

con sede a **Napoli** in **Via Ferrante Imparato, 198**

La formazione si è svolta dal **27/03/2024** al **27/03/2024**

Data **04/04/2024**

  
Il Datore di Lavoro  
(Firma)  
**Teoresi S.p.A.**  
via Perugia, 24 - 10152 Torino  
c.f. e p.iva 03037960014

  
Il docente  
**Costabile Lo Schiavo**