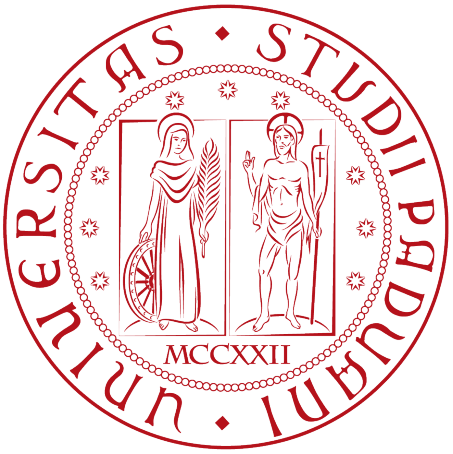
**Università degli Studi di Padova**

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica




# Studio del sistema SAP B1, Analisi della sua Integrazione con Servizi REST e Prototipazione di un Add-on

*Tesi di laurea triennale*

*Relatore*
Prof. Massimiliano De Leoni

*Laureando*
Alberto Crivellari

Anno Accademico 2020-2021

# Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecentoventi ore, dal laureando Alberto Crivellari presso l'Azienda SINAPSI INFORMATICA S.R.L.. Gli obiettivi da raggiungere erano molteplici:

* In primo luogo era richiesto lo studio del gestionale SAP Business One e webservices che interagiscono con esso;
* In secondo luogo era richiesto lo sviluppo di un add-on da applicare sulla maschera di un modulo del SAP:
  - Quest'add-on consiste nell'aggiunta di una funzionalità di stampa di un form del SAP su file esterno oppure stampa come finestra di messaggio, in una finestra all'interno del client.
* Terzo ed ultimo obiettivo era l'ampliamento di parti dei webservices, in particolare 3 diversi ampliamenti:
  - Aggiunta di un campo "idext", rappresentante l'id esterno di un intervento;
  - Aggiunta nel webservice di aggiunta intervento di un campo "extraj_module", rappresentante informazioni aggiuntive sull'intervento, sotto forma di stringa;
  - Aggiunta del campo "extraj_module", anche nel webservice di lettura degli interventi.

Gli obiettivi e risultati raggiunti dall'applicazione sono stati considerati più che sufficienti dall'azienda. In particolare sono stati raggiunti tutti gli obiettivi obbligatori concordati nel piano di lavoro.

Purtroppo non è stato possibile effettuare test automatici del codice, poichè il codice non è abbastanza corposo da renderli necessari.

*"All things are difficult before they are easy"*

— Dr. Thomas Fuller

# Ringraziamenti

*Innanzitutto, vorrei esprimere la mia gratitudine al Prof. De Leoni Massimiliano, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.*

*Desidero ringraziare con affetto i miei genitori e i miei fratelli per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.*

*Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.*

*Padova, Settembre 2021* Alberto Crivellari

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Capitolo 1

# Introduzione

## 1.1 L'azienda

Sinapsi Informatica SRL è una società di sviluppo software che da più di 20 anni si occupa di fornire assistenza ad altre aziende nella gestione d'impresa e ottimizzazione dei processi produttivi. Sinapsi Informatica fornisce ai propri clienti conoscenze e strumenti necessari per il raggiungimento degli obiettivi e degli standard di qualità dei processi interni. L'azienda offre un contratto di assistenza post vendita basato su un ammontare di ore prepagate, solitamente un pacchetto di 100 ore, ma che può cambiare in base alle richieste del cliente. Dunque il principale lavoro dell'azienda consiste in questa assistenza post vendita, come assistenza telefonica, telematica o on-site, ovvero di persona.

### 1.1.1 Principali prodotti

I principali prodotti venduti e installati dall'azienda sono:

1. SAP Business One:
   Gestionale SAP Business One, punto forte dell'azienda che verrà trattato in profondità in questa tesi.

2. SAP Hana:
   Relational DBMS alternativo a MySQLServer, il cui proprietario è SAP, mentre il proprietario di MySQL è Microsoft.
   I database che utilizzano SAP Hana sono esclusivamente su server Linux.

3. Lotus:
   Si presenta con Lotus Domino lato server e Lotus Notes lato client.
   Il client Lotus Notes viene utilizzato da tutto il personale come mail aziendale, calendario aziendale e molte altre funzionalità, tra cui archivio password condiviso e knowledge base interna, dove sono annotate le procedure su problematiche già riscontrate.

4. Sophos:
   Sophos viene utilizzato come antivirus e firewall virtuale.

## 1.2 Descrizione dello stage

L'azienda è stata fin da subito interessata nell'espandere le proprie conoscenze attraverso il contatto con nuove persone, ovvero noi studenti, oltre ad espandere il progetto di stage proposto con protipazione di un add-on e ampliamento di webservices. Il progetto di stage riguarda:

* l'apprendimento del gestionale SAP e dei vari moduli connessi ad esso dall'azienda;
* lo sviluppo di un'applicazione add-on da applicare al client SAP:
  - Quest'add-on consiste nell'aggiunta di una funzionalità di stampa di un form del SAP su file esterno oppure stampa come finestra di messaggio, in una finestra all'interno del client.
* la modifica di alcuni webservices che connettono il client SAP ad un database secondario AWS, in particolare:
  - Aggiunta di un campo "idext", rappresentante l'id esterno di un intervento;
  - Aggiunta nel webservice di aggiunta intervento di un campo "extraj_module", rappresentante informazioni aggiuntive sull'intervento, sotto forma di stringa;
  - Aggiunta del campo "extraj_module", anche nel webservice di lettura degli interventi.

## 1.3 Descrizione capitoli

**Il secondo capitolo** descrive l'architettura software dell'infrastruttura preesistente in azienda.

**Il terzo capitolo** approfondisce la descrizione dello stage, introdotta a grandi linee in questo capitolo introduttivo.

**Il quarto capitolo** descrive analisi e sviluppo dell'add-on.

**Il quinto capitolo** descrive le modifiche effettuate ai webservices in AWS.

**Il sesto capitolo** trae le conclusioni finali.

# Capitolo 2

# Infrastruttura Preesistente e tecnologie

*In questo capitolo verrà descritta l'architettura software del gestionale SAP e dei moduli aggiunti dall'azienda*

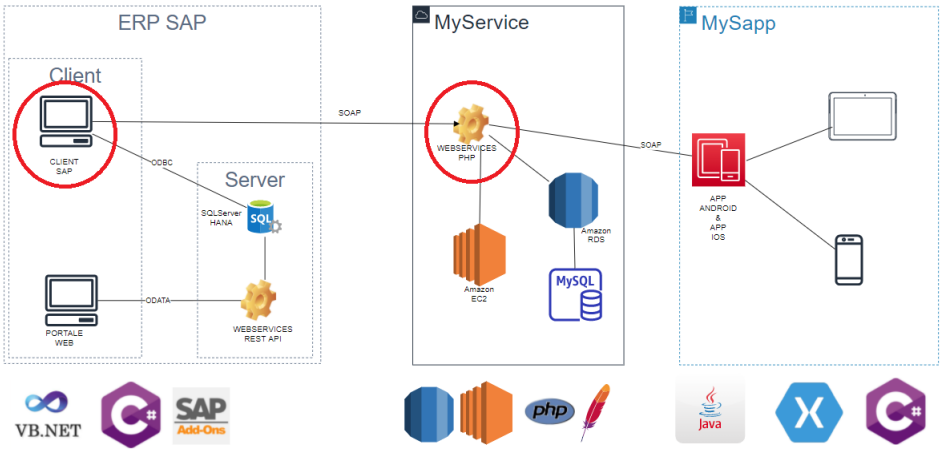



**Figura 2.1:** Schema di rappresentazione modulo gestionale ERP SAP e moduli aggiunti dall'azienda intorno al gestionale

L'azienda ha una infrastruttura per collegare il gestionale a delle applicazioni smartphone ed un portale web.
Si basa su 3 moduli che si connettono tramite protocollo SOAP.
Dalla figura 2.1 possiamo notare i tre moduli principali:

* **ERP SAP:** il modulo del gestionale;
* **MyService:** il modulo dei webservices AWS;
* **MySapp:** il modulo delle applicazioni Android e iOS.

Sulle due parti evidenziate dal cerchio rosso si è focalizzato il lavoro svolto durante lo stage. Nel client SAP verrà applicato l'applicazione add-on, mentre nei webservices php sono state effettuate delle modifiche ad alcune funzioni.

## 2.1 Descrizione modulo ERP SAP

### 2.1.1 ERP SAP

ERP SAP rappresenta il modulo del gestionale, in particolare nel nostro caso abbiamo come gestionale il SAP Business One.
ERP è l'acronimo di Enterprise Resource Planning.
SAP è l'acronimo di Systems, Applications, Products.

SAP è un'azienda leader nel settore di ERP, e i suoi prodotti sono dei gestionali, ovvero dei software ERP.

Il gestionale SAP è un tipo di software che le organizzazioni utilizzano per gestire le attività commerciali quotidiane, come ad esempio contabilità, project management, gestione del rischio e operazioni e gestione della catena di distribuzione.

Come possiamo vedere in Figura 2.2, il SAP è suddiviso in client e server.
Il server può essere basato su :

* **SQLServer:** solo su Windows;
* **HANA:** solo su Linux.

Il client SAP attualmente è disponibile solo su Windows.
Sul client SAP possono essere applicati degli add-on, per modificare i comportamenti della GUI del client in base a com'è programmato l'add-on.

I due gestionali più famosi di SAP sono:

* SAP R/3;
* SAP Business One.

Questo progetto di stage è stato incentrato intorno al SAP Business On, dunque ora entreremo più nel dettaglio su quest'ultimo.

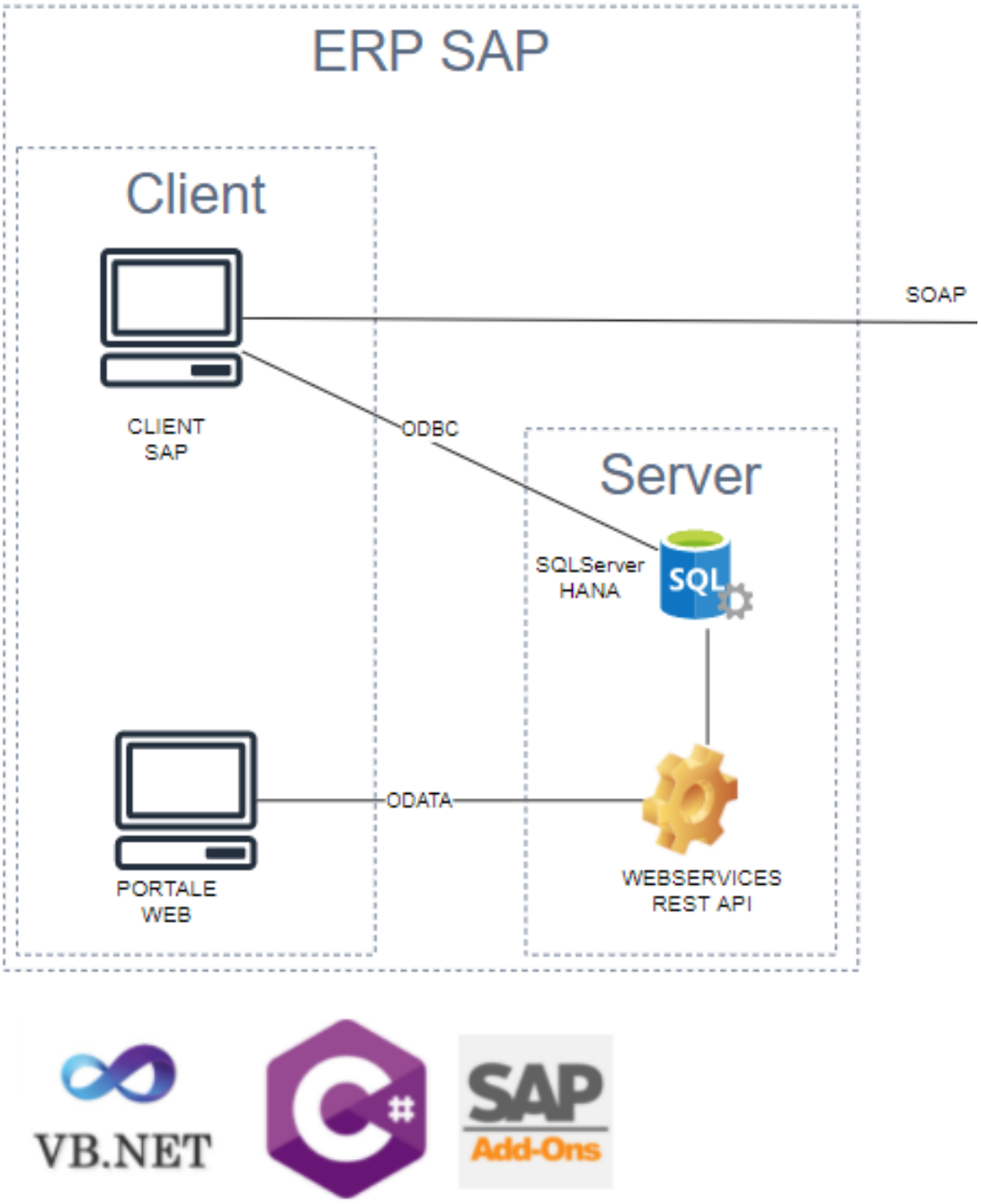



**Figura 2.2:** Schema di rappresentazione modulo gestionale ERP SAP

### 2.1.2 SAP Business One

SAP Business One, abbreviato a SAP B1, è un sofware ERP per le piccole/medie imprese.

Questo gestionale è attualmente alla versione 10, rilasciata a marzo 2020, e rispetto al SAP R/3 consente molte più personalizzazione per essere adatto alle esigenze più disparate.

Come detto in precedenza è un tipico software con modello Client-server.

* Il client è principalmente il Client SAP, o B1 Client, ma può essere anche un portale web oppure un'applicazione mobile;
* Il server viene eseguito su un database Microsoft SQL Server (Windows) oppure un database SAP HANA (Linux).

Infine, come detto in precedenza, SAB B1 consente di effettuare molte personalizzazioni (ovvero gli add-on) utilizzando **SAP Business One SDK**, ovvero un insieme di strumenti e librerie disponibili per lo sviluppo di add-ons su Microsoft Visual Studio, con C# o VB.NET.

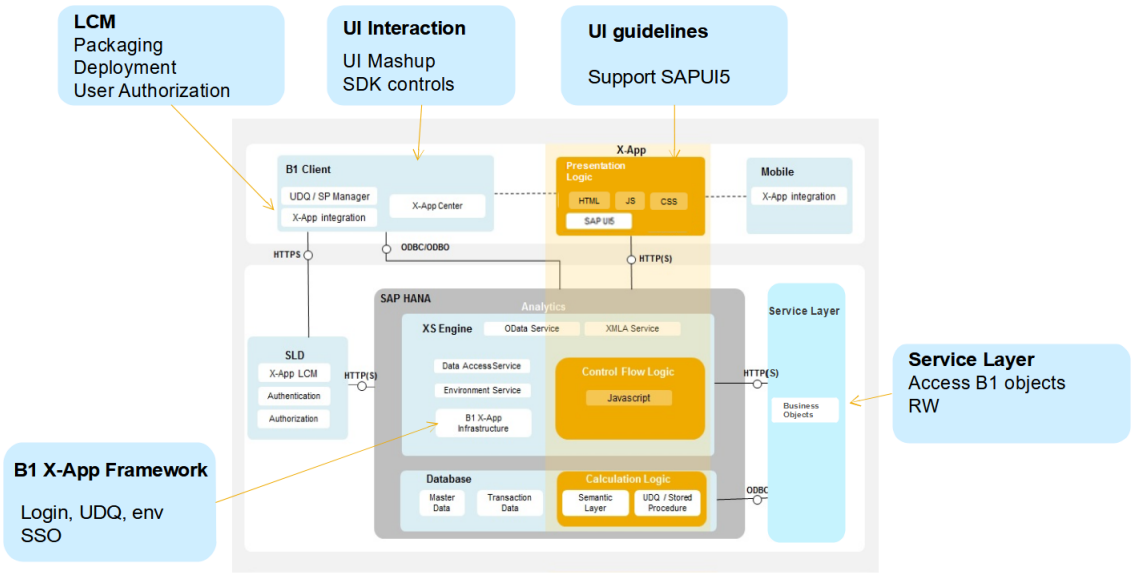



**Figura 2.3:** Schema ufficiale sul SAP Business One

Nella figura 2.3 possiamo vedere uno schema sul SAP Business One, ufficiale dalla documentazione SAP.
Possiamo vedere i vari client con cui accedere al SAP:

* **B1 Client:** ovvero il client SAP desktop;
* **X-APP:** ovvero un portale web sviluppato da SAP;
* **X-APP per mobile:** l'integrazione X-APP, per i dispositivi mobile;
* **Service Layer:** i webservices REST API, che possono essere utilizzati a loro volta da altre applicazioni (ad esempio, sito web o applicazioni mobile), per accedere al SAP.

Tutti questi client accedono al SAP attraverso vie diverse.

* Il client SAP, B1 Client, accede al database attraverso:
  1. Open DataBase Connectivity (ODBC)
  2. Object linking and embedding DataBase for Online analytical processing (ODBO)

  e successivamente accede a SLD, ovvero la componente di autenticazione, via HTTP(S);
* I webservices accedono al server SAP attraverso HTTP(S), e ricevono risposta dal database attraverso ODBC;
* X-APP, sia versione web che mobile, accede al server SAP, e quindi successivamente al database, attraverso HTTP(S).

Sempre nella figura 2.3, si può vedere XS Engine, ovvero il server SAP (la logica SAP) e il database, contenuti dentro il DBMS SAP HANA, ma lo stesso vale se il DBMS è Microsoft SQL Server.
Ora proseguiamo analizzando le componenti principali del SAP, ovvero:

* il client SAP;
* il server SAP;
* il Service Layer di SAP, ovvero i webservices REST API.

### 2.1.3 Client SAP

Il client SAP utilizza la connessione ODBC per comunicare con il server SAP. ODCB, ovvero Open DataBase Connectivity, è uno standard utilizzato per la connessione tra client e DBMS.

Il client SAP utilizza il protocollo SOAP per comunicare con il webserver AWS del modulo MyService.

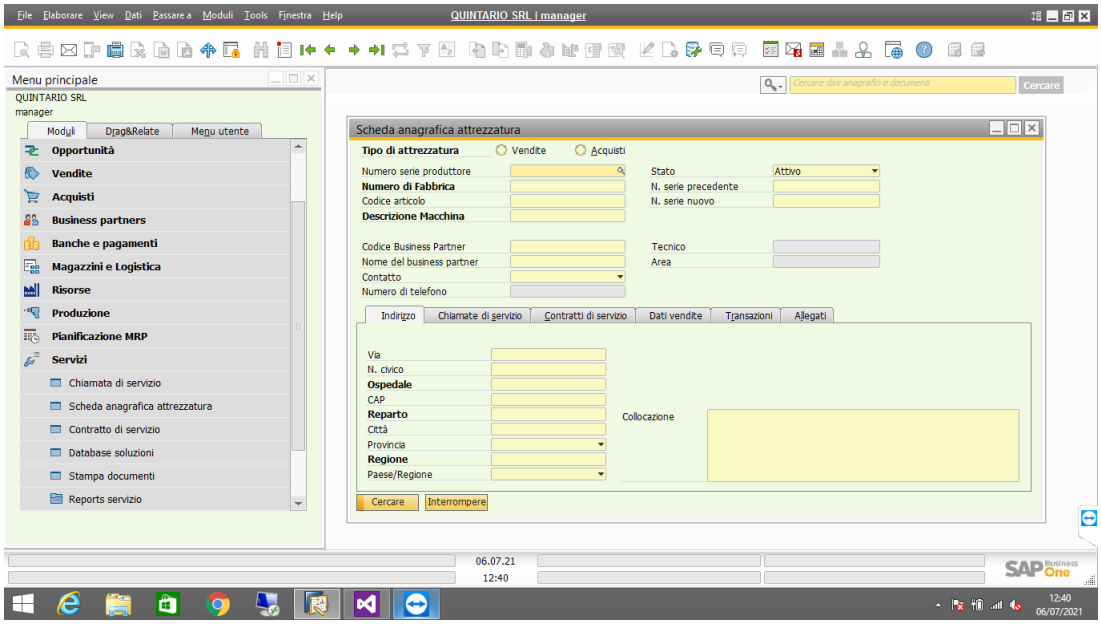



**Figura 2.4:** Client SAP, aperto sulla scheda "Scheda anagrafica attrezzatura"

Nella figura 2.4 possiamo vedere come appare il Client SAP.
In questo caso è stata aperta la Scheda anagrafica attrezzatura, del modulo dei Servizi. Da notare che questi "moduli" del SAP, differiscono dai moduli coinvolti con il modulo del gestionale (MyService e MySapp), questi sono moduli interni del SAP.

La scheda anagrafica attrezzatura è una scheda che rappresenta le attrezzature dell'azienda, ad esempio macchine a controllo numerico, lavatrici o condizionatori, e tutti i possibili macchinari dell'azienda.

Ora vediamo un esempio di cambiamento della GUI di questo client con l'applicazione di un Add-On alla scheda anagrafica attrezzatura.

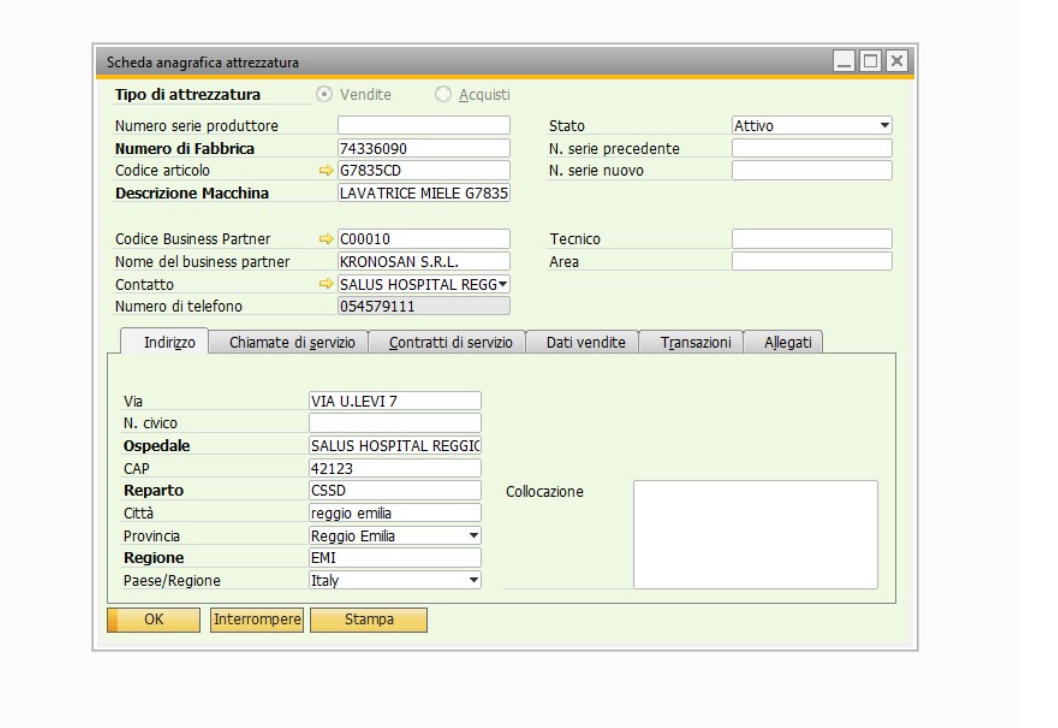



**Figura 2.5:** Client SAP, esempio di modifica GUI di un add-on applicato sulla scheda "Scheda anagrafica attrezzatura"

Come possiamo vedere in figura 2.5 è apparso un nuovo pulsante, che stamperà i dati della scheda.
Questo è un'esempio molto semplice, ma si possono cambiare altre cose, ad esempio aggiungere o rimuovere campi, o aggiungere funzioni ad eventi, ad esempio messaggi di testo che appaiono cliccando dei campi particolari.

Questi add-on possono essere programmati tramite Microsoft Visual Studio, con :

* **C#**, C sharp;
* **VB.NET**, Visual Basic NET.

### 2.1.4 Portale Web

Ora mostriamo il portale web, sviluppato dall'azienda, in concomitanza con un'altra azienda specializzata in siti web (mys).
L'admin del portale, oppure l'admin relativo all'azienda cliente, crea le credenziali per i vari utenti, che saranno i dipendenti delle aziende clienti di Sinapsi.
Dunque arrivati al sito web, bisogna inserire le credenziali nella sezione Login, che apparirà come prima pagina del sito.

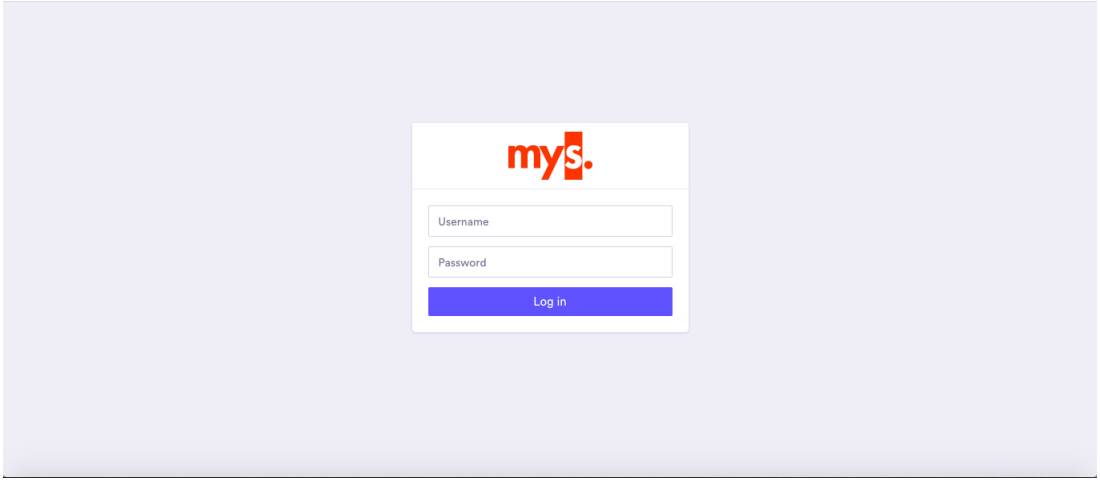



**Figura 2.6:** Portale Web, sezione di Login

Una volta effettuato il login, si viene reindirizzati al sito web, vero e proprio.
Da qui abbiamo principalmente 3 schermate principali:

* Attrezzature;
* Calendario;
* Interventi.

La schermata Attrezzature mostra in forma tabellare tutte le attrezzature e macchinari dell'azienda, con i vari dettagli.

La schermata Interventi mostra sempre in forma tabellare i vari interventi richiesti su quale attrezzatura, e i dettagli dell'intervento.
E' presente un'altra sezione, nella schermata Interventi, la sezione Interventi In Attesa, che mostra gli interventi che si stanno sincronizzando con il database SAP, che non sono ancora sincronizzati.

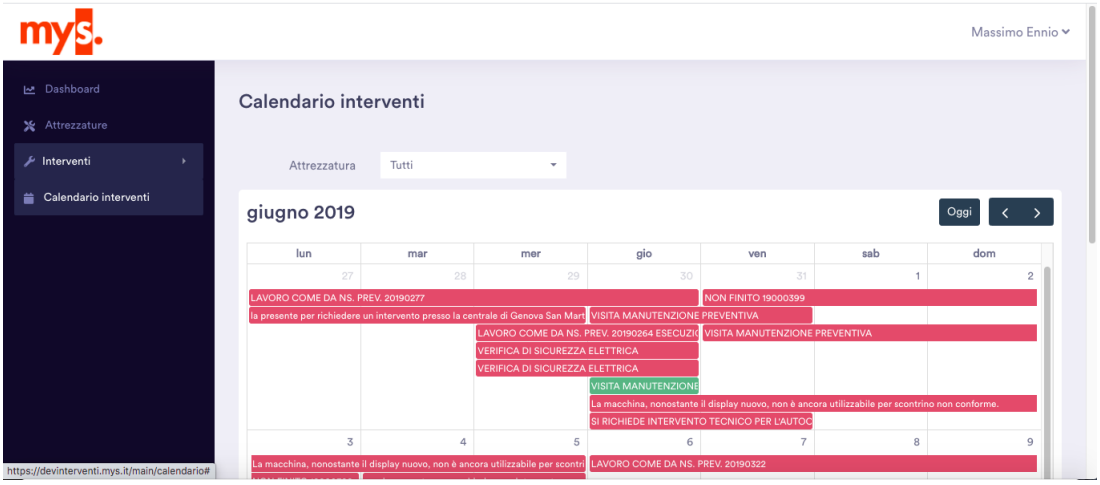



**Figura 2.7:** Portale Web, schermata Calendario

Cominciamo con la schermata Calendario, in figura 2.7.

Evidenziate di rosso ci sono le chiamate di servizio, ad esempio manutenzioni, non completate, mentre evidenziate in verde quelle completate.

Ora continuiamo con la schermata Attrezzature.

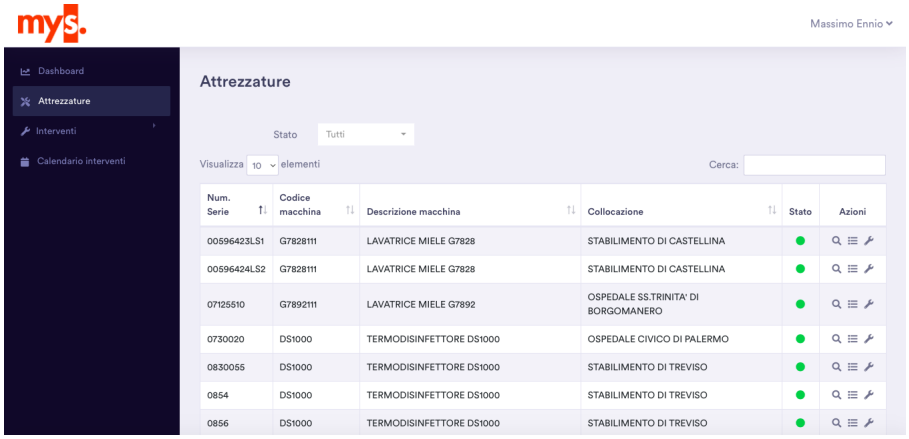



**Figura 2.8:** Portale Web, schermata Attrezzature

Nella figura 2.8 possiamo vedere tutte le attrezzature, relative all'azienda cliente collegata, e quindi alle credenziali.
Per ogni attrezzatura ci sono alcune azioni, tra cui ricerca ed elenco chiamate di servizio.
Tra le varie azioni la più interessante è l'apertura di una nuova chiamata di servizio, ovvero richiedere un nuovo intervento, ad esempio una manutenzione o sostituzione di componenti.

Ora passiamo alla schermata degli interventi, di cui possiamo conoscere tutte le caratteristiche principali, come numero e tipo di intervento, data inizio e fine.
E anche la causale e lo stato, per stato si intende se l'intervento è già stato effettuato.

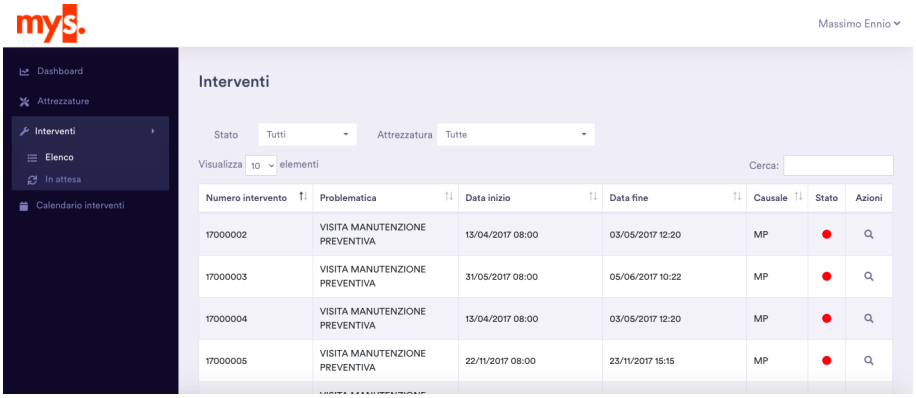



**Figura 2.9:** Portale Web, sezione di Login

Infine, all'interno della schermata degli interventi, abbiamo la sezione "In attesa", per gli interventi in attesa di sincronizzazione.
Questi sono gli interventi creati tramite questo portale web, che si stanno sincronizzando con il server SAP, lo stato verde indica che si sono sincronizzati, quello rosso che non si sono ancora sincronizzati.

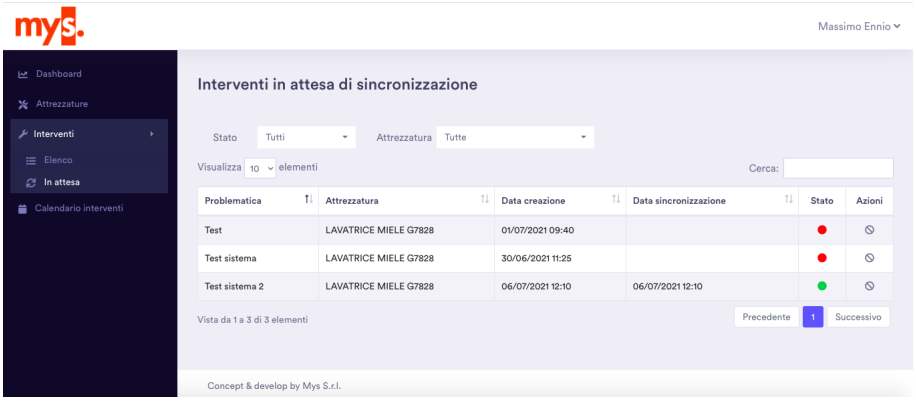



**Figura 2.10:** Portale Web, schermata Interventi, sezione In attesa

### 2.1.5 Webservices REST API

Recentemente, su SAP Business One è stato introdotto un service layer, composto da webservices REST API, che comunicano tramite protocollo ODATA (open data protocol).
Questi webservices permettono di accedere direttamente agli oggetti SAP, senza passare per il client SAP. Gli oggetti SAP sono una struttura dati generata dalla logica del server SAP, che raggruppano e ordinano secondo certe logiche i dati presenti nel database. Per poter ottenere questi dati bisogna fare una richiesta http al webserver, il quale invierà un file json con gli oggetti SAP richiesti.
L'utilizzo più comune di questi webservices è un portale web il quale utilizza le informazioni prese tramite queste richieste HTTP.

APIs format

The URI and content of HTTP request and response follow OData protocol
The payload follows JSON format.

| Operation | URI | Request Content | Response Content on success | Response Content on failure |
|---|---|---|---|---|
| Login | POST http://localhost/b1s/Login | Object JSON | Object JSON | Object JSON |
| Logout | GET/POST http://localhost/b1s/Logout | (Empty) | (Empty) | Object JSON |
| Add | POST http://localhost/b1s/Items | Object JSON | Object JSON | Object JSON |
| Get | GET http://localhost/b1s/Items('I101') | (Empty) | Object JSON | Object JSON |
| Update | PATCH http://localhost/b1s/Items('I101') | Object JSON | {} | Object JSON |
| Delete | DELETE http://localhost/b1s/Items('I101') | (Empty) | {} | Object JSON |

©

**Figura 2.11:** Formato delle richieste HTTP per il Service Layer SAP

Come si può osservare in figura 2.11, sono presenti le richieste HTTP principali per interagire con i webservices REST API del Service Layer di SAP.

A seguito di una richiesta HTTP viene elaborata la richiesta.
Viene restituito un oggetto JSON in caso di fallimento, comunicando l'errore causante il fallimento.
In caso di successo viene restituito un oggetto JSON, ove necessario, oppure un messaggio vuoto che rappresenta il successo dell'operazione.

## 2.2 Descrizione modulo MyService

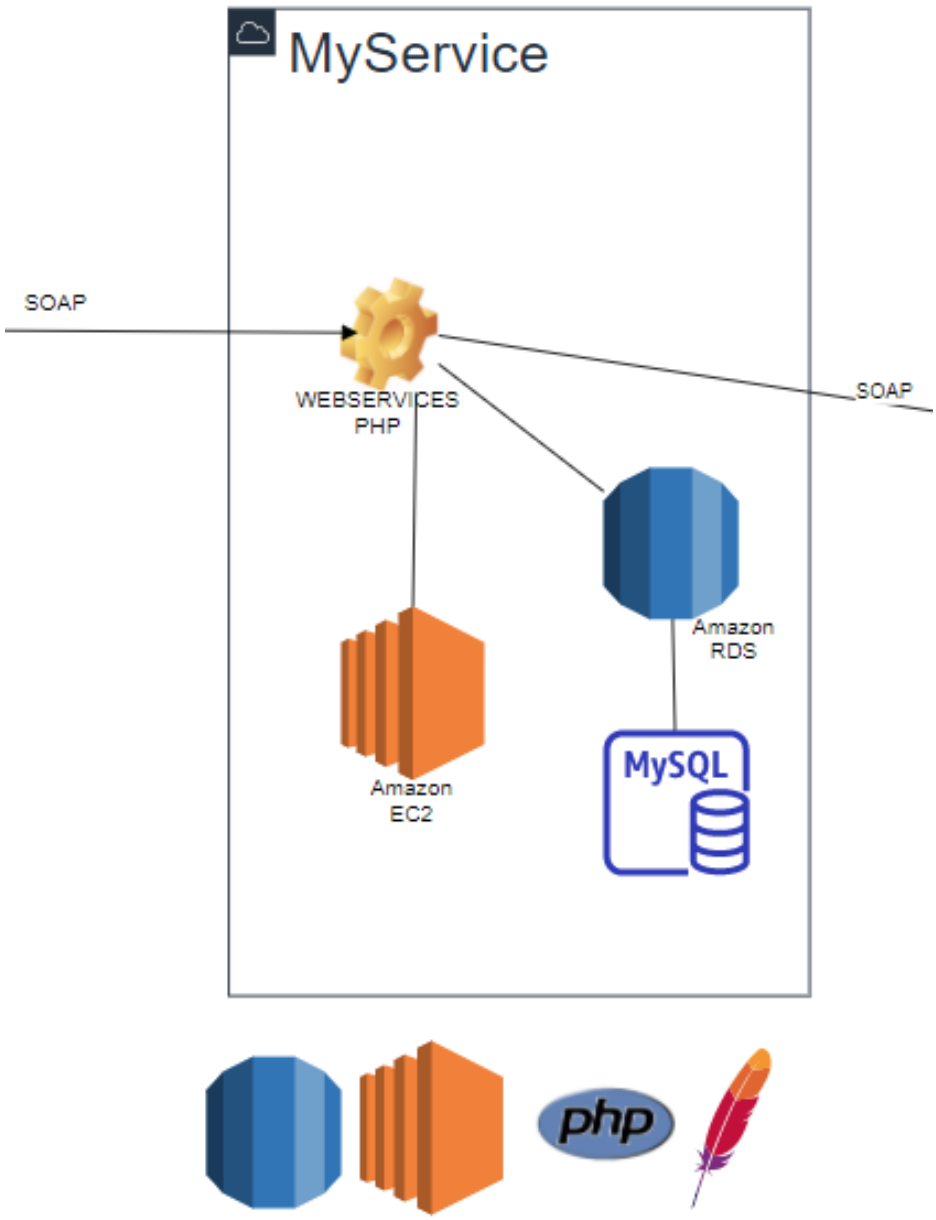



**Figura 2.12:** Schema di rappresentazione modulo MyService dei servizi AWS

### 2.2.1 Servizi AWS

In questo modulo abbiamo i webservices in PHP installati in un server AWS, accompagnato da un database MySQL. AWS è un acronimo che sta per Amazon Web Services. In particolare, dei servizi AWS, in questo modulo abbiamo:

* Amazon AWS Elastic Compute Cloud (EC2), Amazon Elastic Compute Cloud, che consiste nel webserver, ovvero il server dove sono presenti i nostri webservices in php;
* Amazon AWS Relational Database Service System (RDS), Amazon Relational Database Service, che si occupa della gestione del database MySQL;
* Amazon AWS Simple Storage System (S3), Amazon Simple Storage Service, che nello schema non è presente, ma si occupa della gestione di allegati e file troppo pesanti nel database per essere gestiti da Amazon RDS.

Ora rivolgiamo la nostra attenzione ai webservices PHP, che mettono in relazione questo webserver e il database con il modulo SAP e il modulo MySapp, delle applicazioni mobile, attraverso il protocollo SOAP.

### 2.2.2 Webservices PHP

Come possiamo vedere dalle figure 2.11 e 2.1, il client SAP comunica con questi webservices PHP, attraverso un add-on che utilizza il protocollo SOAP.
Per programmare questi webservices sono state utilizzate le tecnologie Apache e PHP.
Il protocollo SOAP, ovvero Simple Object Access Protocol, è un protocollo per lo scambio di messaggi tra componenti software.

I webservices php comunicano anche con le applicazioni mobile, per Android e iOS, sempre tramite protocollo SOAP.
Vi sono moltissime funzioni in PHP che corrispondono a molte funzioni SOAP nei webservices, il cui compito principale è mettere in relazione le app mobile o il SAP, con il database contenuto in questo modulo.
L'obiettivo principale è gestire le chiamate di servizio, ovvero richieste di intervento, ai tecnici del SAP.
Elenchiamo alcune delle funzioni più comuni:

* Visione di tutti i dettagli di una chiamata di servizio;
* Visione di tutte le chiamate di servizio di un tecnico;
* Rimozione di una chiamata di servizio;
* Modifica di una chiamata di servizio;
* Aggiunta di una chiamata di servizio;
* Invio mail a cliente, una volta concluso una chiamata di servizio;
* Login di un tecnico;
* Logout di un tecnico.

## 2.3 Descrizione modulo MySapp

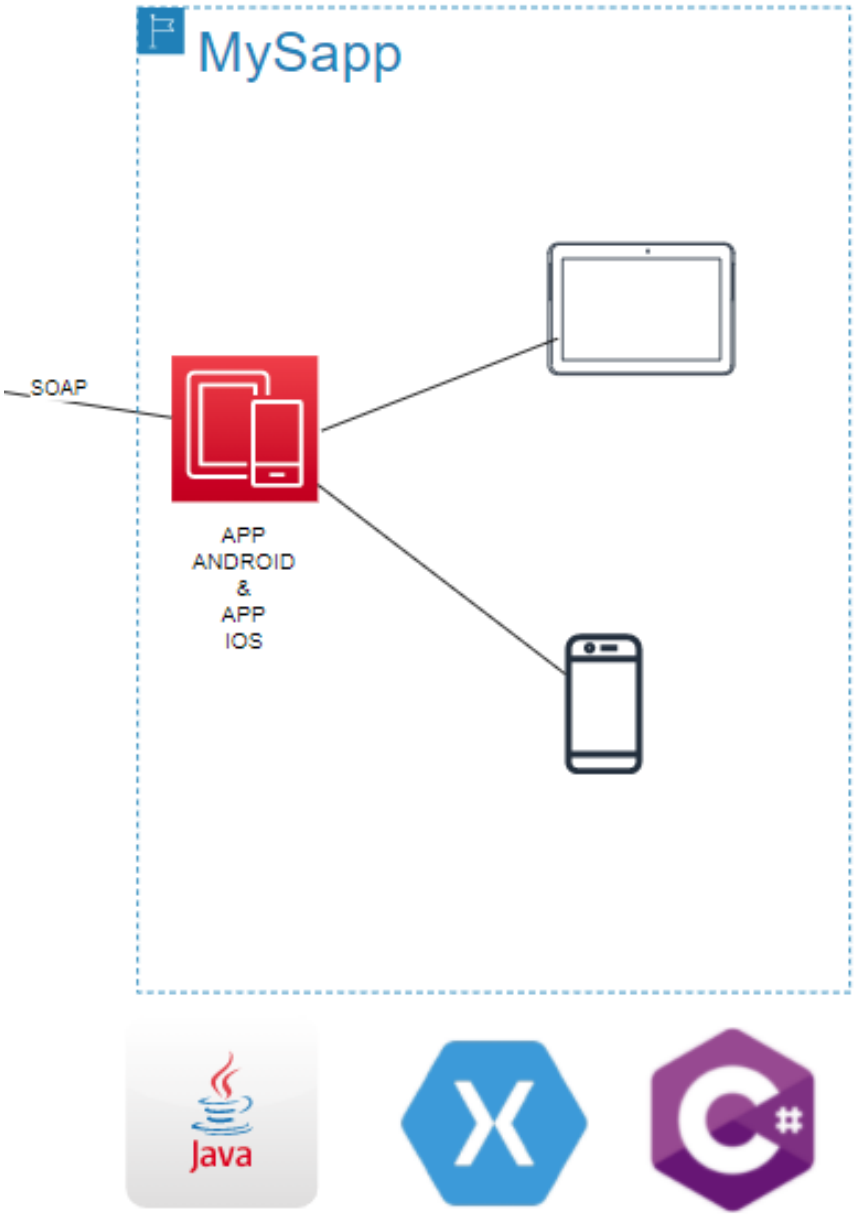



**Figura 2.13:** Schema di rappresentazione modulo MySapp delle applicazioni mobili

### 2.3.1 Applicazioni mobile

Il modulo MyService si collega con questo modulo MySapp, attraverso il protocollo SOAP utilizzando i webservices PHP del modulo MyService.
Sono state sviluppate due applicazioni per i clienti che risolvono le chiamate di servizio on-site, ovvero di persona.
In queste applicazioni, il tecnico esegue il login, con le sue credenziali e successivamente può gestire le chiamate di servizio a lui assegnate.
Ora vediamo velocemente com'è stato impostato il layout delle due applicazioni.

#### Applicazione iOS

Per lo sviluppo dell'applicazione iOS è stato utilizza C# e Xamarin.

Cominciamo con la schermata di login.

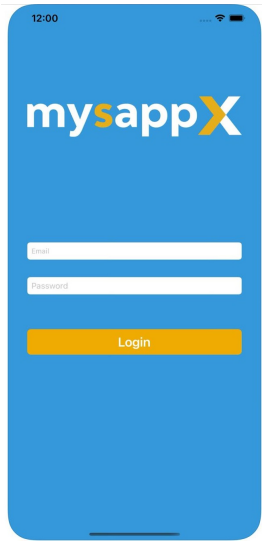



**Figura 2.14:** App iOS, schermata di Login

In figura 2.14 il tecnico dovrà inserire le proprie credenziali e accederà alla gestione delle chiamate di servizio a lui assegnate.

Continuiamo con la schermata Principale dell'app.
In questa schermata si possono vedere le chiamate di servizio assegnate a questo tecnico ancora da eseguire.

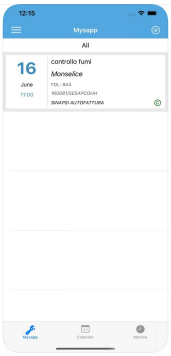



**Figura 2.15:** App iOS, schermata Principale

In figura 2.15 possiamo vedere il calendario e la posizione della o delle chiamate di servizio da eseguire.

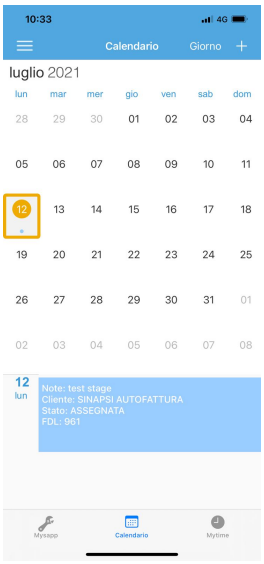



**Figura 2.16:** App iOS, schermata Calendario


**Figura 2.17:** App iOS, schermata Intervento

Infine, in figura 2.17 abbiamo i dettagli della chiamata di servizio selezionata, dove possiamo vedere i vari dettagli della chiamata di servizio, ad esempio:

* posizione del cliente;
* stato dell'intervento;
* attrezzatura che necessita dell'intervento;
* e altri dettagli meno importanti.

### Applicazione Android

Per lo sviluppo dell'applicazione Android è stato utilizzato Java e Kotlin.

Cominciamo a esplorare l'applicazione dalla schermata di login.
In figura 2.18 il tecnico dovrà inserire le proprie credenziali.

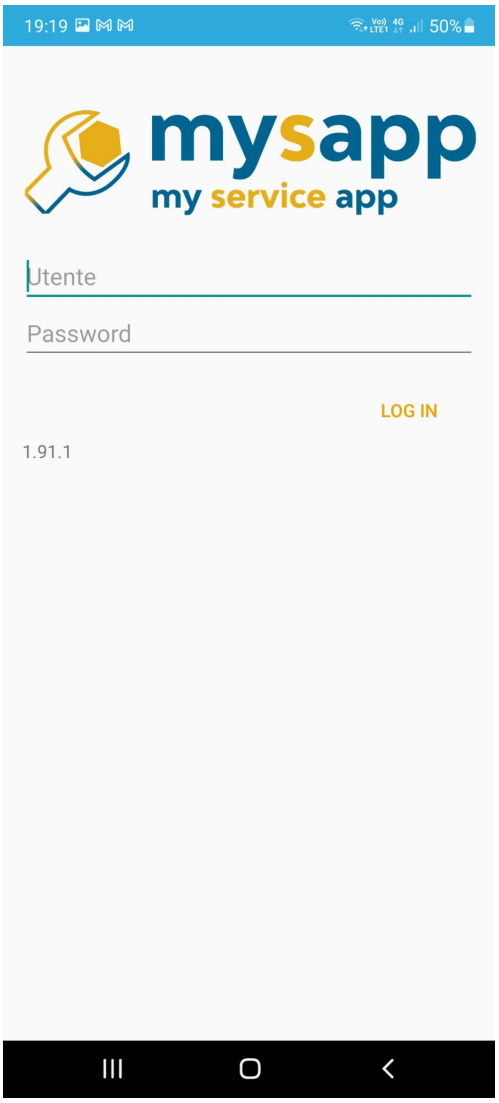



**Figura 2.18:** App Android, schermata di Login

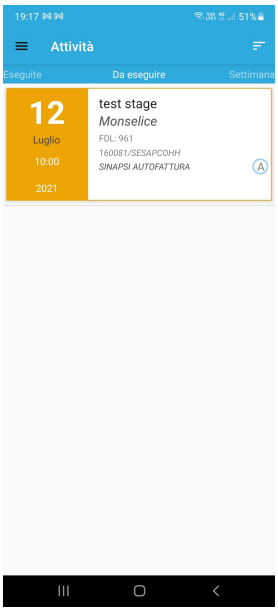



**Figura 2.19:** App Android, schermata Principale

In figura 2.19 possiamo vedere la schermata Principale dell'app, dove si possono vedere le chiamate assegnate al tecnico che ha eseguito l'accesso nell'applicazione. In

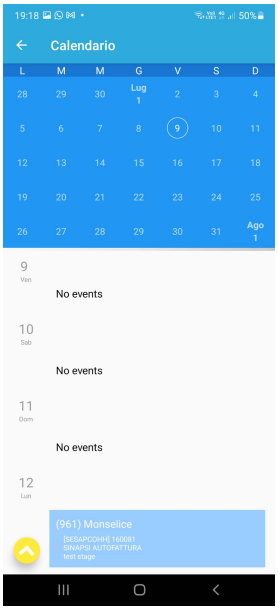



**Figura 2.20:** App Android, schermata Calendario

figura 2.20 possiamo vedere la schermata calendario dell'app.

Infine, abbiamo i dettagli della chiamata di servizio selezionata.

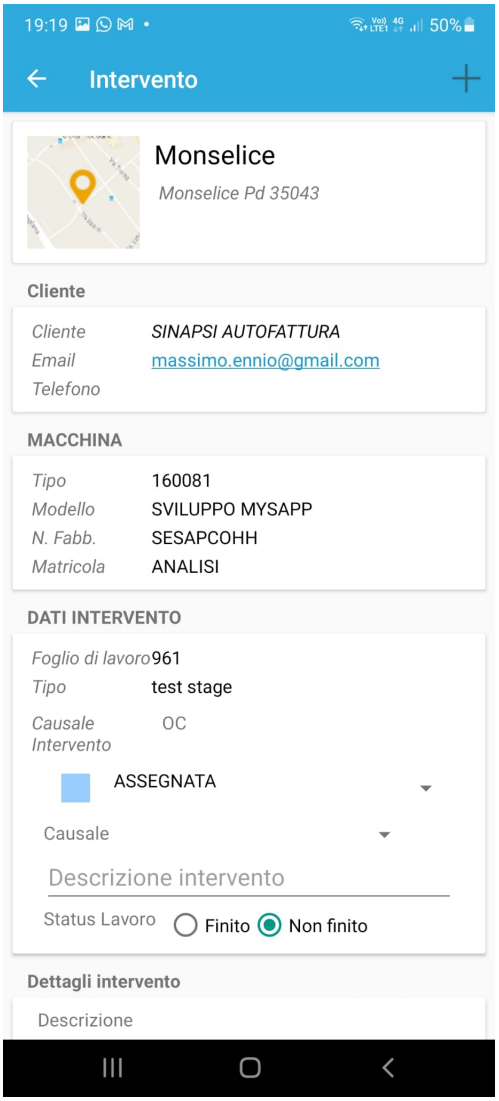



**Figura 2.21:** App Android, schermata Intervento

In figura 2.21 possiamo vedere i vari dettagli della chiamata di servizio, ad esempio:

- posizione del cliente;
- attrezzatura che necessita dell'intervento;
- dati e dettagli dell'intervento.

## 2.4 Tecnologie coinvolte

In questa sezione approfondiremo le tecnologie coinvolte durante lo stage:

* **PHP:** Insieme ad Apache viene utilizzato per sviluppare applicazioni web lato server.

  In questo caso, viene utilizzato dall'azienda per lo sviluppo e gestione dei webservice sui server AWS.

* **VB.NET:** Linguaggio di programmazione per programmare gli add-ons, su Visual Studio, utilizzando le librerie SAP Business One (SAPB1).

* **C#:** Linguaggio di programmazione alternativo per programmare gli add-ons.

  Sempre su Visual Studio, utilizzando le librerie fornite dalla SAPB1 SDK.

* **SoapUI:** Applicazione utilizzata per testare ed interagire con i webservices PHP, con protocollo SOAP.

  E' possibile inserire una chiamata e l'applicazione ritorna una risposta dopo aver interagito con i webservice.

* **HeidiSQL:** Strumento di gestione di databases, quali MySQL, MariaDB, InnoDB.

  In questo caso utilizzato per il database MySQL del server AWS, per controllare che le operazioni di webservices modifichino o meno il database.

* **MIcrosoft Visual Studio:** Strumento di sviluppo software, Integrated Development Environment (IDE), utilizzato per programmare codice di vario tipo, da applicazione desktop ad app mobile, e raramente utilizzato per applicazioni web.

  In questo caso necessario per programmare gli add-ons, poichè le librerie necessarie sono predisposte per Visual Studio.

* **Microsoft SQL Server Management Studio (SSMS):** Applicazione utilizzata per gestire i database, viene utilizzata particolarmente per i server SQL Server.

  Nel nostro caso l'applicazione viene utilizzata per accedere direttamente ai database del server SAP.

* **Remote Desktop:** Applicazione preinstallata su Windows per accedere ad altri computer nella stessa rete locale.

  Viene utilizzata per accedere ai server SAP, presenti nella rete locale aziendale.

* **Domino Lotus Notes:** Applicazione utilizzata principalmente per la mail aziendale.

  Contiene però altre funzionalità quali un calendario aziendale e un server condiviso (domino) aziendale, che comprende database di fatture e password per accedere a diversi software aziendali o di clienti dell'azienda.

* **Postman:** Applicazione utilizzata per gestire e testare le API.

  Nel caso dell'azienda è stata utilizzata principalmente per controllare il corretto funzionamento e testare nuovi tipi di richieste sui webservices REST API di SAPB1.

# Capitolo 3

# Descrizione dello stage

*In questo capitolo descriveremo gli obiettivi e il lavoro effettuato nello stage a grandi linee, per poi approfondire nel dettaglio nei prossimi capitoli*

## 3.1 Introduzione al progetto

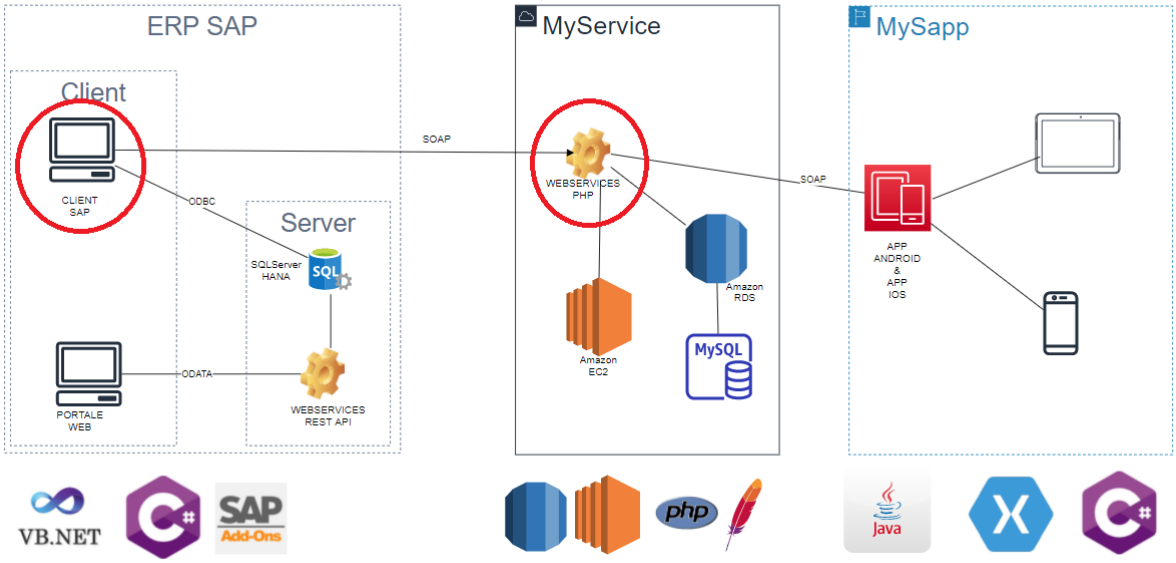



**Figura 3.1:** Schema infrastruttura preesistente, con evidenziati le componenti obiettivo del lavoro di stage

In figura 3.1 possiamo vedere lo schema dell'infrastruttura preesistente, già vista nel capitolo precedente.
Evidenziate in rosso, con un cerchio, possiamo vedere le due componenti su cui è stato concentrato il lavoro durante questo stage:

* **Client SAP:** è stato creato un nuovo add-on, utilizzabile dal client SAP;
* **Webservices PHP:** sono state modificate ed ampliate alcune funzioni che compongono i webservices in PHP.

### 3.1.1 Vincoli tecnologici

L'azienda ha posto due vincoli tecnologici:

* nello sviluppo dell'add-on, utilizzare VB.NET oppure C#;
* negli ampliamenti dei webservices solamente il linguaggio PHP.

Il vincolo sugli add-on è tale, poichè quest'ultimi possono essere programmati soltanto in quei due linguaggi.

Il vincolo sugli ampliamenti dei webservices è dettato dal dover modificare delle funzioni in PHP, dunque bisogna utilizzare lo stesso linguaggio.

## 3.2 Interazione

Per lo svolgimento dell'attività di stage si è deciso in comune accordo con il tutor aziendale di svolgere lo stage completamente in azienda.
Questo è stato deciso per favorire il dialogo tra studente e tutor aziendale, ed eventualmente un confronto con gli altri dipendenti dell'azienda, esperti nel settore d'interesse.

## 3.3 Requisiti e obiettivi

Sin dal piano di lavoro sono stati decisi alcuni obiettivi, di cui 3 obbligatori e 1 desiderabile.
Obiettivi obbligatori:

* Comprensione e apprendimento di strumenti di programmazione e infrastruttura esistente;
* Sviluppo protocolli di comunicazione tra i diversi moduli;
* Sviluppo applicazione add-on.

Obiettivo desiderabile:

* Test e documentazione.

### 3.3.1 Raggiungimento obiettivi

Gli obiettivi e risultati raggiunti dall'applicazione sono stati considerati più che sufficienti dall'azienda. In particolare sono stati raggiunti tutti gli obiettivi obbligatori concordati nel piano di lavoro.

Purtroppo non è stato possibile effettuare test automatici del codice, poichè il codice non è abbastanza corposo da renderli necessari.

## 3.4 Pianificazione

La durata dello stage è stimata attorno alle 320 ore circa.
Per ottenere una buona organizzazione dell'attività di stage, è stato stilato il piano di lavoro assieme al tutor aziendale Massimo Ennio e al relatore Massimiliano De Leoni.

### 3.4.1 Piano di Lavoro

La ripartizione delle ore è stata distribuita nelle 8 settimane in questo modo:

| Durata in ore | Descrizione dell'attività |
|---|---|
| 20 | Configurazione ambiente di lavoro |
| 40 | Formazione sulle varie tecnologie |
| 60 | Formazione su infrastruttura esistente e servizi AWS |
| 60 | Sviluppo protocolli di comunicazione per l'integrazione dei moduli |
| 80 | Sviluppo applicazione add-on |
| 30 | Test e documentazione |
| 30 | Finalizzazione e discussione risultati |
| Totale ore | |
| 320 | |

**Figura 3.2:** Ripartizione ore, secondo il piano di lavoro, durante l'attività di stage

# Capitolo 4

# Sviluppo Add-On

*In questo capitolo approfondiremo lo sviluppo dell'add-on*

## 4.1 Analisi dei requisiti

### 4.1.1 Introduzione

Fulcro dell'attività di stage è stata la produzione e sviluppo di un add-on, un'applicazione a maschera, applicabile al client SAP.
Lo scopo di quest'add-on è la stampa di un form o tabella di un modulo di SAP, ad esempio il form Scheda Attrezzatura, oppure l'esportazione su file esterno.

Quest'applicazione è stata codificata in Microsoft Visual Studio, con le librerie di SAP Business One, in linguaggio C#.
Per realizzarlo c'è stato un periodo di formazione personale sul linguaggio C# e sulle librerie di SAP per lo sviluppo degli add-on.

Continuiamo con la presentazione dei casi d'uso e il tracciamento dei requisiti.

### 4.1.2 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi.
I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

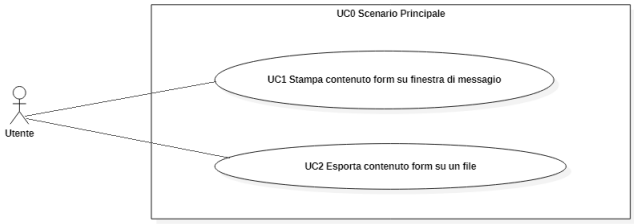



**Figura 4.1:** Use Case - UC0: Scenario principale

## UC0: Scenario principale

**Precondizioni:** L'utente è entrato in un form del SAP.

**Descrizione:** L'add-on mette a disposizione le funzionalità di stampa o salvataggio su file, dei contenuti del form.

**Postcondizioni:** Il sistema è pronto per permettere una nuova interazione.

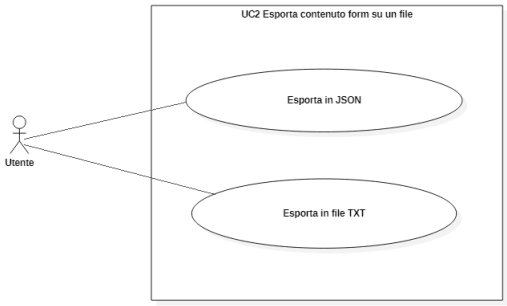



**Figura 4.2:** Use Case - UC2: Esporta contenuto form su un file

## UC2: Esporta contenuto form su un file

**Precondizioni:** L'utente ha selezionato di salvare i contenuti del form su un file esterno.

**Descrizione:** L'add-on metta a disposizione il salvataggio su file .txt o su file .json.

**Postcondizioni:** Il sistema è pronto per permettere una nuova interazione.

### 4.1.3 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.

Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.

Il codice dei requisiti è così strutturato R(F/Q/V)(O/D/F) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

O = obbligatorio

D = desiderabile

F = facoltativo

Nelle tabelle 4.1 e 4.2 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

| Requisito | Descrizione | Use Case |
|---|---|---|
| RFO-1 | L'add-on deve permettere di stampare il contenuto del form sotto forma di finestra di messaggio | UC1 |
| RFO-2 | L'add-on deve permettere di esportare il contenuto del form in un file in formato JSON | UC2.1 |
| RFO-3 | L'add-on deve permettere di esportare il contenuto del form in un file in formato TXT | UC2.2 |

**Tabella 4.1:** Tabella del tracciamento dei requisiti funzionali

| Requisito | Descrizione | Use Case |
|---|---|---|
| RVO-1 | La versione di SAP Business One da utilizzare è la versione 10.0 | - |

**Tabella 4.2:** Tabella del tracciamento dei requisiti di vincolo

## 4.2 Realizzazione

Lo sviluppo dell'add-on è stato effettuato, come detto in precedenza, con linguaggio C# e attraverso l'IDE Microsoft Visual Studio.
Inoltre è stata usata una delle varie librerie fornite da SAP Business One SDK, ovvero:

* **SAPbouiCOM:** ovvero SAP Business One UI COM, una libreria COM di SAP Business One, per la gestione dell'interfaccia grafica(*User Interface*).

Nella sezione precedente, non sono stati forniti diagrammi di classe poichè il codice dell'add-on, consiste in una classe che gestisce gli eventi generali ed una classe di supporto che gestisce gli eventi del form aperto.
Dunque, essendo solo due classi, è stato ritenuto poco utile fornire diagrammi.

Nel proseguo, viene illustrata una realizzazione tramite uno scenario di utilizzo.

Per prima cosa accediamo al SAP con le credenziali corrette ed entriamo su un form, di un modulo.
In questo caso entriamo nel form Scheda Attrezzatura, mostrato in figura 4.3.

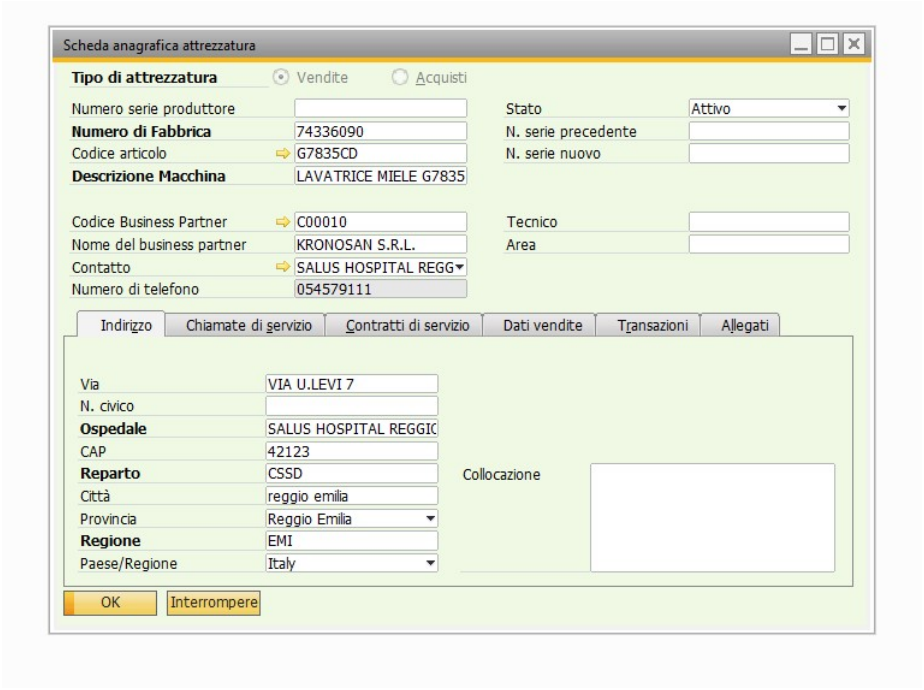



**Figura 4.3:** Add-On, screenshot prima dell'attivazione dell'add-on

Come possiamo notare in figura 4.3 abbiamo due pulsanti:

* Ok;
* Interrompere.

Ora attiviamo l'add-on:

* installare l'add-on nel client SAP locale;
* compilare e farlo eseguire su Microsoft Visual Studio, senza averlo installato nel client SAP;
* nel caso sia installato nel client SAP, si può impostare l'attivazione manuale o automatica;
* nel caso di attivazione manuale, bisogna andare sulle opzioni degli add-on installati ed attivarlo.

Una volta attivato l'add-on, riapriamo il modulo Scheda Attrezzatura e vedremo comparire un pulsante nuovo.

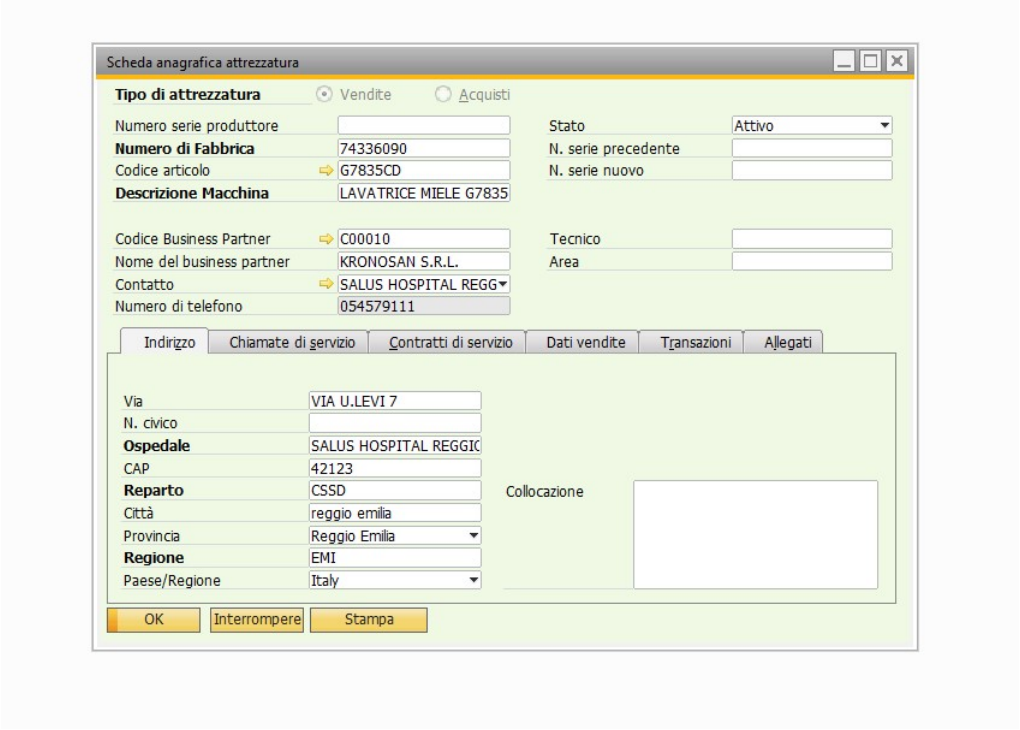



**Figura 4.4:** Add-On, screenshot dopo l'attivazione dell'add-on

In figura 4.4 è già stata selezionata un'attrezzatura, in questo caso l'ultima attrezzatura aggiunta nel database.

Cliccando il pulsante Stampa si aprirà una nuova finestra.

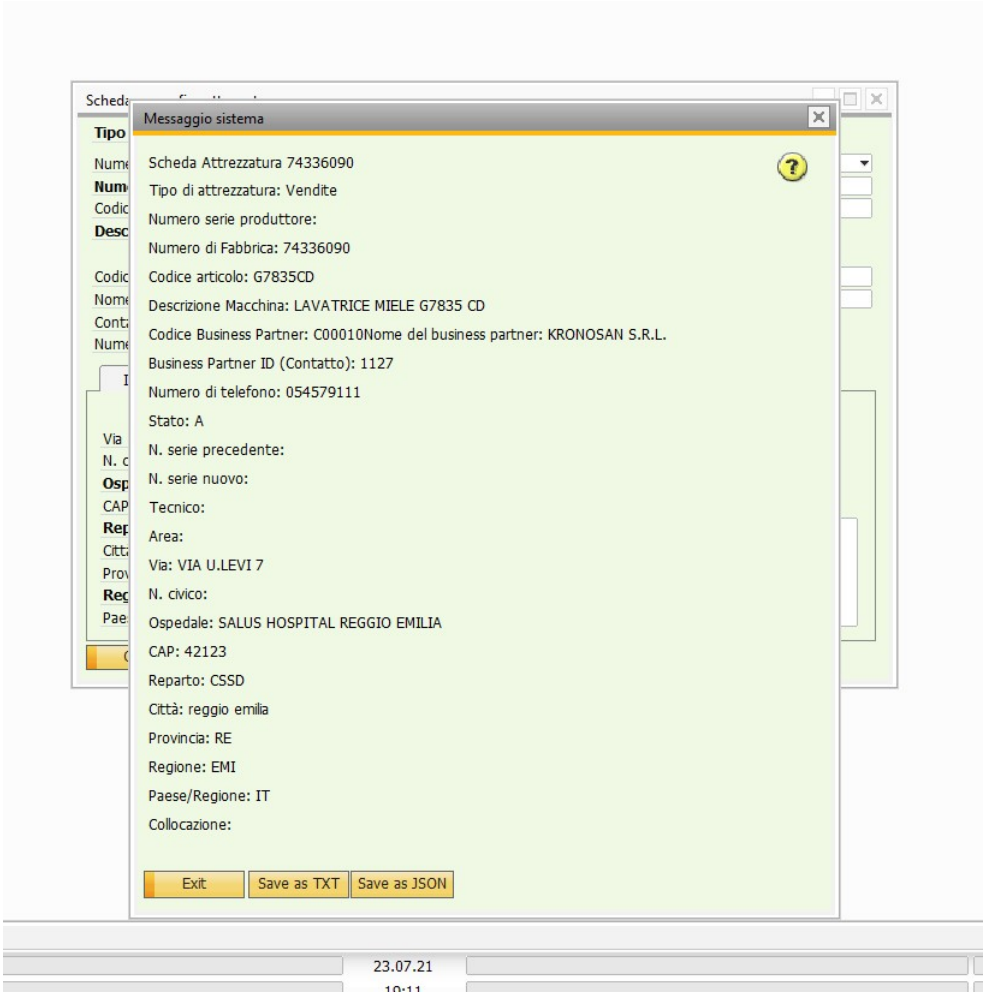



**Figura 4.5:** Add-On, stampa dei contenuti del form su finestra di messaggio

Dopo aver cliccato il pulsante Stampa, si entra nella finestra mostrata in figura 4.5, dove abbiamo stampato su finestra di messaggio i contenuti del form; inoltre abbiamo altri 3 pulsanti:

* **Exit:** premendo questo pulsante, come dice il nome, usciremo da questa finestra;
* **Save as TXT:** premendo questo pulsante il contenuto di questa finestra verrà esportato su un file .txt, senza alcuna formattazione aggiuntiva;
* **Save as JSON:** premendo questo pulsante il contenuto di questa finestra verrà esportato su un file .json, con la formattazione json.

I file vengono salvati nella cartella dov'è installato l'add-on, in particolare vengono chiamati rispettivamente **file.txt** per il file TXT e **filejson.json** per il file JSON. Se questi file esistono già nella cartella, e viene premuto nuovamente il pulsante di salvataggio, i file verranno sovrascritti dall'ultimo file.

Ora nei prossimi due screenshot, vedremo il contenuto dei due file, rispettivamente txt e json, esportati dall'add-on.

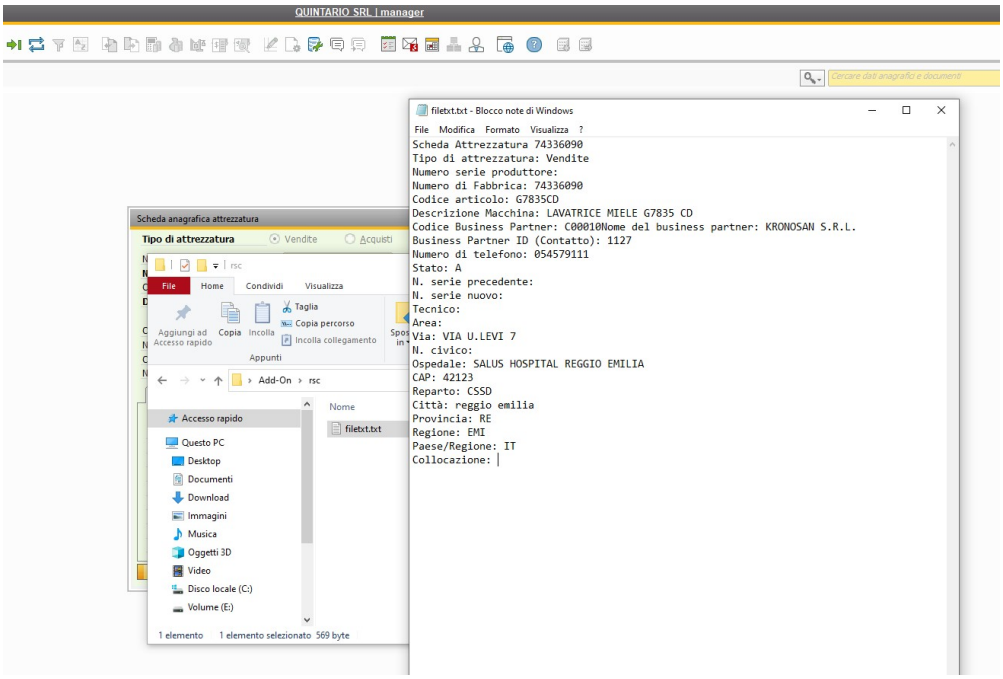



**Figura 4.6:** Add-On, file txt esportato dall'add-on

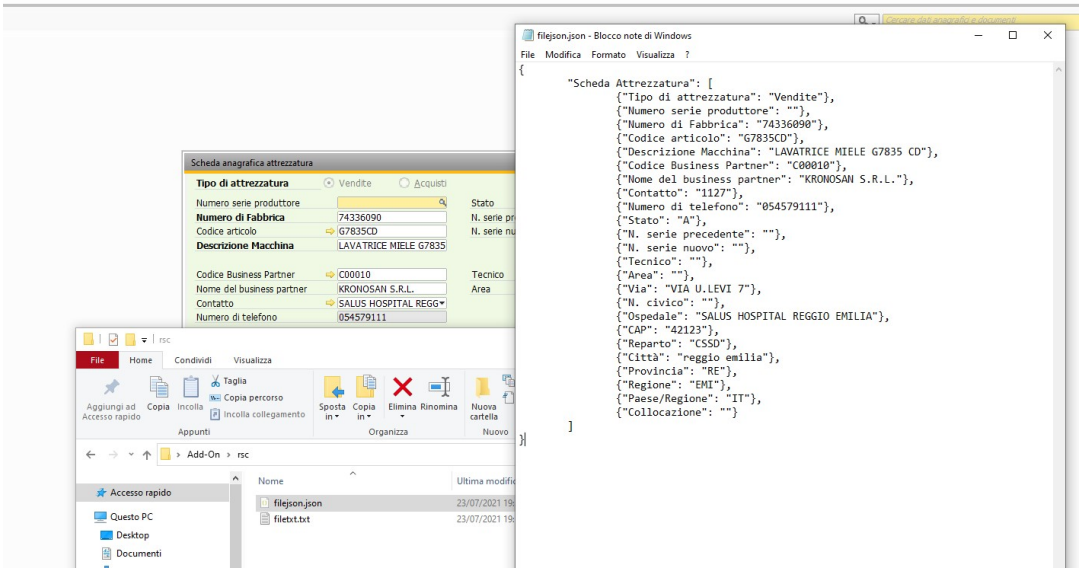



**Figura 4.7:** Add-On, file json esportato dall'add-on

## 4.3 Accettazione

Riguardo la Verifica e Validazione, ovvero l'Accettazione, non abbiamo alcun test d'unità e test automatico. Dunque per verificare l'add-on abbiamo una sequenza di operazioni da eseguire per verificarne il corretto funzionamento.

La procedura per verificare il corretto funzionamento dell'applicazione è la seguente:

1. Accedere al SAP Business One con le proprie credenziali;
2. Attivare l'add-on dalle opzioni del client;
3. Aprire il form Scheda Attrezzatura nel modulo dei Servizi;
4. Selezionare un'istanza di attrezzatura esistente, usualmente attraverso il numero di fabbrica, essendo chiave univoca di questa tabella;
5. Cliccare il pulsante Stampa;
6. Verificare che le informazioni stampate nella nuova finestra di messaggio apertasi corrispondano alle informazioni del form;
7. Cliccare Save as JSON;
8. Cliccare Save as TXT;
9. Entrare nella cartella dell'add-on dove sono salvati i file;
10. Verificare che il contenuto dei file corrisponda alle informazioni del form.

Se non ci sono stati problemi in questi step, in particolare se le informazioni stampate nella finestra di messaggio e le informazioni salvate nei due file corrispondono alle informazioni presenti nel form Scheda anagrafica attrezzatura, l'add-on può considerarsi verificato e validato.

### 4.3.1 Requisiti soddisfatti

Logicamente i requisiti funzionali sono stati tutti soddisfatti, eccone qui una piccola tabella.

| **Requisito** | **Descrizione** | **Use Case** |
|---|---|---|
| RFO-1 | Step 6 | Soddisfatto |
| RFO-2 | Step 10 | Soddisfatto |
| RFO-3 | Step 10 | Soddisfatto |

**Tabella 4.3:** Tabella del soddisfacimento dei requisiti funzionali

# Capitolo 5

# Ampliamento webservices PHP

*In questo capitolo approfondiremo gli ampliamenti dei webservices PHP*

## 5.1 Introduzione

Altra componente dell'attività di stage sono i 3 ampliamenti dei webservices PHP, presenti nel server AWS, nel modulo MyServices. Oltre a modificare i webservices è stato modificata anche la struttura del database, sempre nel server AWS, che interagisce con i webservices.

Questi webservices sono presenti sotto forma di vari file PHP, con molte funzioni che interagiscono fra di loro.

Nel corso dello stage si è lavorato su questi ampliamenti dei webservices PHP preesistenti.

Sono 3 ampliamenti, ora li elenchiamo brevemente:

1. Aggiunta di un campo "idext", rappresentante l'id esterno di un intervento;
2. Aggiunta nel webservice di aggiunta intervento di un campo "extraj_module", rappresentante informazioni aggiuntive sull'intervento, sotto forma di stringa;
3. Aggiunta del campo "extraj_module", anche nel webservice di lettura degli interventi.

Ora approfondiamo più nel dettaglio questi ampliamenti.

## 5.2 Ampliamenti webservices

### 5.2.1 Primo ampliamento

Il primo ampliamento riguarda la modifica del webservice:
**MokersSyncGetActivities**.

In particolare l'ampliamento riguarda l'aggiunta di un campo idext nella funzione che restituisce una o più chiamate di servizio e data una chiamata di servizio ritorna tutti i suoi dettagli.
Tra questi dettagli dobbiamo aggiungere il dettaglio di idext, ovvero id esterno, la chiave esterna della tabella delle chiamate di servizio nel database.

```
2354 /**
2355  * MokersSyncGetActivities: per ottenere gli interventi.
2356  */
2357 $server->wsdl->addComplexType(
2358     'requestMoKersSyncGetActivities',
2359     'complexType',
2360     'struct',
2361     'all',
2362     '',
2363     array(
2364         'begin' => array('name' => 'begin', 'type' => 'xsd:dateTime', 'minOccurs' => '0', 'maxOccurs' => '1'),
2365         'end' => array('name' => 'end', 'type' => 'xsd:dateTime', 'minOccurs' => '0', 'maxOccurs' => '1'),
2366         'laststates' => array('name' => 'laststates', 'type' => 'tns:mokersStatesList', 'minOccurs' => '0', 'maxOccurs' => '1'),
2367         'synced' => array('name' => 'synced', 'type' => 'xsd:boolean', 'minOccurs' => '0', 'maxOccurs' => '1'),
2368         'enabled' => array('name' => 'enabled', 'type' => 'xsd:boolean', 'minOccurs' => '0', 'maxOccurs' => '1'),
2369         'withAttachments' => array('name' => 'withAttachments', 'type' => 'xsd:boolean', 'minOccurs' => '0', 'maxOccurs' => '1'),
2370+        'idext' => array('name' => 'idext', 'type' => 'xsd:int', 'minOccurs' => '0', 'maxOccurs' => '1')
2371     )
2372 );
```

**Figura 5.1:** Aggiunta idext tra i parametri di ritorno del webservice

Nella figura 5.1 possiamo vedere la funzione di ritorno in WSDL del webservice **MokersSyncGetActivities**, che ritorna i dettagli di tutte le chiamate di servizio (ovvero le Activities) richieste.
Questa funzione legge nel database le chiamate di servizio e ritorna un array contenente tutti i dettagli, a questo array di dettagli abbiamo aggiunto come ultimo campo, il campo idext.

Nelle figure 5.2 e 5.3 due screenshot relativi a questo ampliamento, di implementazione interna della funzione PHP.

```
2603 /**
2604  * Classe che definisce i filtri per la selezione di Attività.
2605  */
2606 class MokerFilters
2607 {
2608     public $beginPlanned; //DateTime
2609     public $endPlanned; //DateTime
2610     public $states; //array di stati
2611     public $synced; //se sincronizzato (true) o meno (false), null indifferente.
2612     public $enabled; //true se abilitato o meno (false), null indifferente.
2613     public $completed; //true se completato o meno (false), null indifferente. è un parametro specifico per cisa
2614     public $withAttachments;
2615+    public $idext;
2616
2617     public function __construct()
2618     {
```

**Figura 5.2:** Aggiunta idext tra i parametri dell'array interno

```
2217     if (array_key_exists("synced", $filters) && isset($filters["synced"])) {
2218         if ($filters["synced"] == true) {
2219             $getFilters->synced = true;
2220         } else if ($filters["synced"] == false) {
2221             $getFilters->synced = false;
2222         }
2223
2224     }
2225
2226+    if (array_key_exists("idext", $filters) && isset($filters["idext"])) {
2227+            $getFilters->idext = $filters["idext"];
2228+    }
2229+
2230     if (array_key_exists("enabled", $filters) && isset($filters["enabled"])) {
2231         if ($filters["enabled"] == true) {
2232             $getFilters->enabled = true;
2233         } else if ($filters["enabled"] == false) {
2234             $getFilters->enabled = false;
2235         }
2236
2237     }
```

**Figura 5.3:** Trascrizione idext dall'array interno all'array di ritorno

In queste funzioni per il webservice **MokersSyncGetActivities** viene utilizzato un array interno, di implementazione, e un array di ritorno.
Nelle figure 5.2 e 5.3 possiamo notare:

* Aggiunta di idext nella struttura dell'array interno (classe MokersFilters);
* Trascrizione di idext, nel caso esista nell'array interno, dall'array interno all'array di ritorno.

Questo è tutto per il primo ampliamento, ora proseguiamo con il secondo.

### 5.2.2 Secondo ampliamento

Il secondo ampliamento riguarda la modifica del webservice:
**MokersSyncNewActivity**.

In particolare l'ampliamento riguarda l'aggiunta di un nuovo campo extraj_module, come stringa.
Quando si crea un nuovo intervento, o chiamata di servizio, si può aggiungere tra i dettagli dell'intervento questo campo extraj_module.
In figura 5.4 e 5.5 abbiamo gli screenshot che mostrano le modifiche effettuate alle funzioni del webservice di creazione di un nuovo intervento.

```
2149  * MokersSyncNewActivity: per l'aggiunta di un nuovo intervento.
2150  */
2151 $server->wsdl->addComplexType(
2152     'requestMoKersSyncNewActivity',
2153     'complexType',
2154     'struct',
2155     'all',
2156     '',
2157     array(
2158         'workerMail' => array('name' => 'workerMail', 'type' => 'xsd:string'),
2159         'idext' => array('name' => 'idext', 'type' => 'xsd:string'),
2160         'rag_soc' => array('name' => 'rag_soc', 'type' => 'xsd:string'),
2161         'presso' => array('name' => 'presso', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
2162         'nome' => array('name' => 'nome', 'type' => 'xsd:string'),
2163         'cognome' => array('name' => 'cognome', 'type' => 'xsd:string'),
2164         'indirizzo' => array('name' => 'indirizzo', 'type' => 'xsd:string'),
2165         'citta' => array('name' => 'citta', 'type' => 'xsd:string'),
2166         'cap' => array('name' => 'cap', 'type' => 'xsd:string'),
2167         'provincia' => array('name' => 'provincia', 'type' => 'xsd:string'),
2168         'telefono1' => array('name' => 'telefono1', 'type' => 'xsd:string'),
2169         'telefono2' => array('name' => 'telefono2', 'type' => 'xsd:string'),
2170         'motivo' => array('name' => 'motivo', 'type' => 'xsd:string'),
2171         'extraj' => array('name' => 'extraj', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
2172+        'extraj_module' => array('name' => 'extraj_module', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
2173         'inizio_pian' => array('name' => 'inizio_pian', 'type' => 'xsd:dateTime'),
2174         'push' => array('name' => 'push', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
2175     )
2176 );
2177
```

**Figura 5.4:** Aggiunta extraj_module tra i parametri in input del webservice

In figura 5.4 possiamo vedere la funzione in WSDL del webservice **MokersSyncNewActivity**, che crea una chiamata di servizio(Activity) ed inserisce i vari dettagli.
Questa funzione scrive nel database le chiamate di servizio in una nuova riga, ovvero una nuova voce della tabella del database.


**Figura 5.5:** Trascrizione idext dall'array interno all'array di ritorno

Lo screenshot in figura 5.5 invece, legge dagli input del webservice i parametri, e se extraj_module esiste, lo aggiunge nell'array interno **$taskData**.


**Figura 5.6:** Aggiunta extraj_module nella query INSERT INTO

In figura 5.6 possiamo vedere l'aggiunta di extraj_module e il suo valore, se l'array interno della chiamata di servizio contiene un campo extraj_module non vuoto.

Ora mostriamo come la query INSERT INTO si sta creando, per far capire cosa significa il precedente screen sull'aggiunta a concatenazione di extraj_module e il suo valore.

```
 */
function insertTask($idazien, $taskData, $attachedFiles = null, MokerActivityDetails $activityDetails = null)
{
    $id_itecs = null; //l'inserimento di un nuovo intervento fornisce un iditecs

    //transaction
    $can_do_commit = true;
    begin();

    if (!empty($activityDetails) && property_exists($activityDetails, "activityType")) {
        $taskData->tipo = $activityDetails->activityType;
    }

    $query_intervento = "INSERT INTO `itecs_mokers` SET ";
    $query_intervento .= "`fk_idazien` = " . mysql_real_escape_string($idazien) . ", ";
    if (!empty($taskData->tipo)) {
```

**Figura 5.7:** Query INSERT INTO che si sta formando

Dalla figura 5.7 si può capire come il webservice crei la query **$query_intervento**, che è una query di tipo INSERT INTO.
Questa query inserisce la chiamata di servizio salvata nell'array interno **$taskData** nella tabella delle chiamate di servizio.

Proseguiamo con il prossimo ampliamento.

### 5.2.3 Terzo ampliamento

Il terzo ampliamento riguarda la modifica del webservice: **MokersSyncGetActivities**.

In particolare l'ampliamento riguarda l'aggiunta di un campo extraj_module nella funzione che data una chiamata di servizio restituisce tutti i dettagli.
Tra questi dettagli dobbiamo aggiungere il dettaglio di extraj_module, ovvero il campo stringa inserito nel secondo ampliamento.
Semplicemente dobbiamo rendere questo webservice in grado di restituire il campo extraj_module se è presente nella voce richiesta delle chiamate di servizio.

```
1521
1522 $server->wsdl->addComplexType(
1523     'payloadMoKers',
1524     'complexType',
1525     'struct',
1526     'all',
1527     '',
1528     array(
1529         'data_inizio_pian' => array('name' => 'data_inizio_pian', 'type' => 'xsd:dateTime', 'nillable' => 'true'),
1530         'data_fine_pian' => array('name' => 'data_fine_pian', 'type' => 'xsd:dateTime', 'nillable' => 'true'),
1531         'idext' => array('name' => 'idext', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
1532         'id_cliente' => array('name' => 'id_cliente', 'type' => 'xsd:int', 'nillable' => 'true'),
1533         'indirizzo' => array('name' => 'indirizzo', 'type' => 'xsd:string', 'nillable' => 'true'),
1534         'rag_soc' => array('name' => 'rag_soc', 'type' => 'xsd:string', 'nillable' => 'true'),
1535         'presso' => array('name' => 'presso', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
1536         'nome' => array('name' => 'nome', 'type' => 'xsd:string', 'nillable' => 'true'),
1537         'cognome' => array('name' => 'cognome', 'type' => 'xsd:string', 'nillable' => 'true'),
1538         'telefono1' => array('name' => 'telefono1', 'type' => 'xsd:string', 'nillable' => 'true'),
1539         'telefono2' => array('name' => 'telefono2', 'type' => 'xsd:string', 'nillable' => 'true'),
1540         'lat' => array('name' => 'lat', 'type' => 'xsd:string', 'nillable' => 'true'),
1541         'lng' => array('name' => 'lng', 'type' => 'xsd:string', 'nillable' => 'true'),
1542         'riscontro' => array('name' => 'riscontro', 'type' => 'xsd:string', 'nillable' => 'true'),
1543         'id_causale' => array('name' => 'id_causale', 'type' => 'xsd:int', 'nillable' => 'true', 'minOccurs' => '0', 'maxOccurs' => '1'),
1544         'motivo' => array('name' => 'motivo', 'type' => 'xsd:string', 'nillable' => 'true'),
1545         'extraj' => array('name' => 'extraj', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
1546+        'extraj_module' => array('name' => 'extraj_module', 'type' => 'xsd:string', 'minOccurs' => '0', 'maxOccurs' => '1'),
```

**Figura 5.8:** Aggiunta extraj_module tra i parametri di ritorno del webservice

In figura 5.8 possiamo vedere la funzione di ritorno in WSDL del webservice **MokersSyncGetActivities**, che restituisce i dettagli di tutte le chiamate di servizio richieste.

Questa funzione legge nel database le chiamate di servizio e ritorna un array contenente tutti i dettagli, a questo array di dettagli abbiamo aggiunto come altro campo di ritorno il campo extraj_module.

Questo webservice si chiama payloadMokers, come si può vedere nell'immagine alla riga 1523, ed è contenuto dentro al webservices MokersSyncGetActivites.

Esso rappresenta il payload di una chiamata di servizio, mentre nel webservice principale possono venire restituite una o più chiamate, in base ai valori dati in input al webservice.

Ora mostriamo com'è la query SELECT, per far capire come si crea l'array di ritorno.

```
$select_intervento = "
SELECT itecs_mokers.*, itecs_mokers_causali.des_causale, list_driver.*
FROM ((itecs_mokers
LEFT JOIN list_driver ON itecs_mokers.fk_cod_driver = list_driver.cod_driver AND itecs_mokers.fk_idazien = list_driver.idazien)
LEFT JOIN itecs_mokers_causali ON itecs_mokers.fk_id_causale = itecs_mokers_causali.id)
WHERE itecs_mokers.enable = 1 AND list_driver.cod_driver = $cod_driver AND list_driver.idazien = $idazien";
$result_intervento = qdb($select_intervento);

if (row($result_intervento) > 0) {
    // ottengo un array con tutti i campi dell'intervento estratti dal database
    $singolo_intervento = from_db_result_to_array($result_intervento);
```

**Figura 5.9:** Query SELECT da cui vengono presi i dati delle chiamate di servizio per comporre l'array di ritorno

In figura 5.9 possiamo vedere **$singolo_intervento** che è una variabile che contiene una chiamata di servizio, mentre **$result_intervento** contiene tutte le chiamate di servizio.

In base agli input dati al webservice, vengono filtrate le chiamate richieste e vengono salvati i dati richiesti in un array di ritorno.

```
924   $intervento["payload"]['extraj'] = $singolo_intervento[$i]["extraj"];
925+  $intervento["payload"]['extraj_module'] = $singolo_intervento[$i]["extraj_module"];
926   $intervento["payload"]['priorita'] = $singolo_intervento[$i]["priorita"];
```

**Figura 5.10:** Trascrizione valore extraj_module dalla query all'array di ritorno

Dalla figura 5.11 possiamo vedere che l'array di ritorno è una variabile array bidimensionale **$intervento**.

Inoltre possiamo vedere la trascrizione del valore di extraj_module dall'array interno di implementazione, nell'array di ritorno del webservice.
Con questo ampliamente abbiamo semplicemente aggiunto il campo extraj_module tra i campi trascritti nell'array di ritorno.
Prima dell'ampliamento la funzione del webservice l'avrebbe letto, ma non l'avrebbe trascritto nell'array di ritorno e quindi non l'avrebbe inviato nell'array ritornato.

Ora proseguiamo con la parte di Verifica e Validazione, che consiste nel controllare che il webservice funzioni, provando a inviare richieste attraverso SoapUI.

## 5.3 Verifica e Validazione

Per verificare che gli ampliamenti siano funzionanti sono stati verificati e validati attraverso un controllo di richiesta/risposta del webservice.

Questo controllo è stato eseguito attraverso un'applicazione, chiamata SoapUI, che ci permette di interagire con i webservices PHP con protocollo SOAP.
Per testarli dunque viene effettuata una richiesta SOAP inserendo gli appositi parametri di input, e attraverso SoapUI che contatta i webservice abbiamo una risposta in protocollo SOAP.

Ora andremo a vedere per ogni ampliamento la risposta SOAP dei webservice. Il primo ampliamento viene verificato insieme al terzo, essendo due ampliamenti allo stesso webservice.

### 5.3.1 Accettazione secondo ampliamento

Iniziamo con l'accettazione del secondo ampliamento.

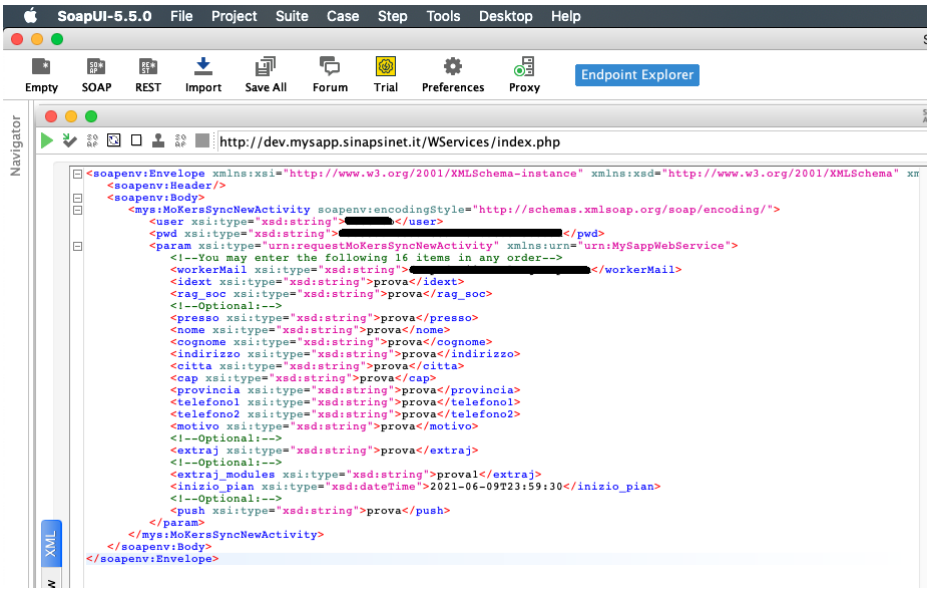



**Figura 5.11:** Secondo Ampliamento webservices, Chiamata SOAP prima dell'ampliamento

Dati aziendali quali username, password e mail del tecnico sono stati censurati.
In questa richiesta SOAP viene inserita una nuova chiamata di servizio e tra i dettagli si aggiunge un campo extraj_module con valore "prova".

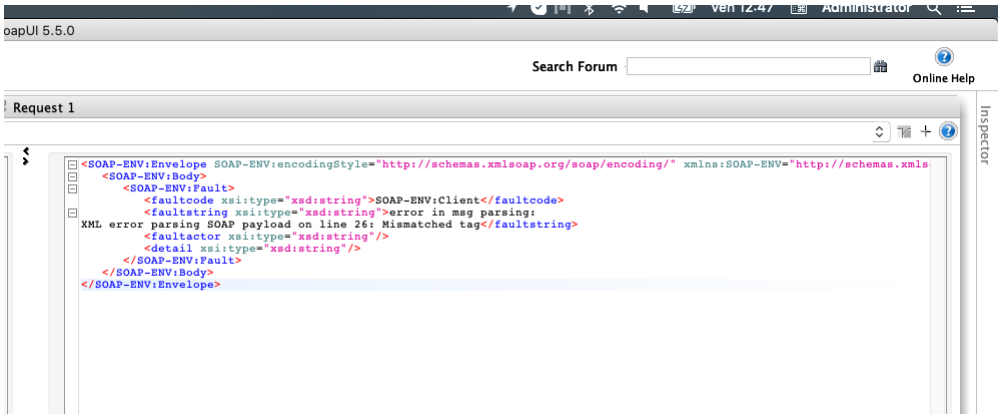



**Figura 5.12:** Secondo Ampliamento webservices, Risposta SOAP prima dell'ampliamento

Se proviamo a dare in input il valore di campo extraj_module si presenta un errore nella risposta.
Prima dell'ampliamento il webservice non si aspetta di ricevere questo input e dunque presenta questo errore di parsing.

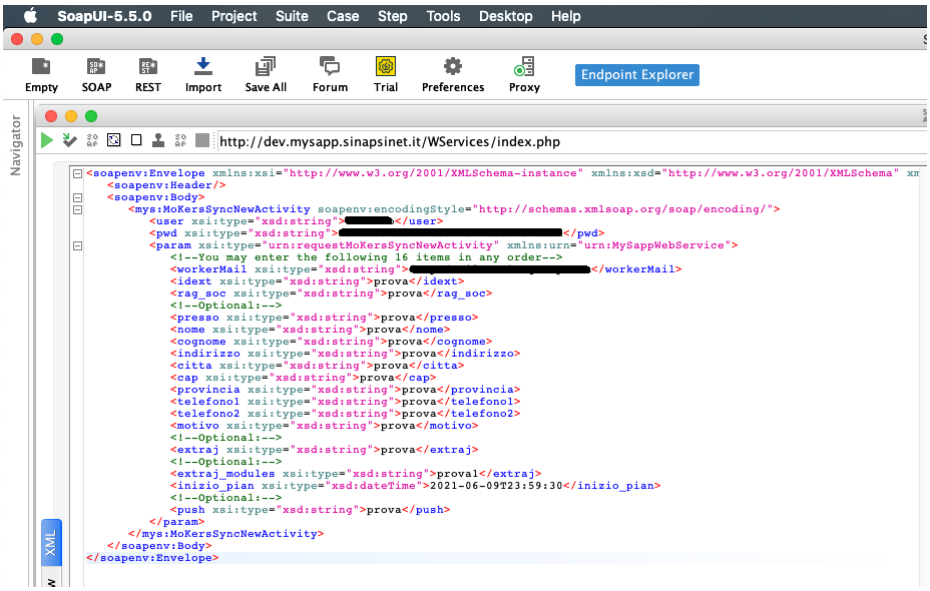



**Figura 5.13:** Secondo Ampliamento webservices, Chiamata SOAP dopo l'ampliamento

Rifacciamo la stessa chiamata della figura 5.12, ma questa volta dopo aver concluso e applicato l'ampliamento.

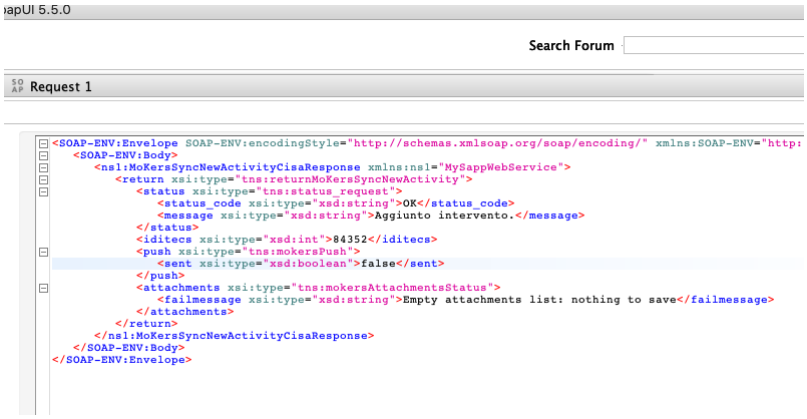



**Figura 5.14:** Secondo Ampliamento webservices, Risposta SOAP dopo l'ampliamento

Questa volta non c'è nessun errore, anzi abbiamo una conferma del successo:

* OK come status_code;
* Aggiunto intervento come message.

Quindi il campo extraj_module viene accettato e inserito nella nuova riga del database insieme agli altri dettagli della chiamata indicati nella richiesta SOAP.

### 5.3.2 Accettazione primo e terzo ampliamento

Proseguiamo con primo e terzo ampliamento.
Essendo entrambi sul webservice MokersSyncGetActivities, vengono verificati dalla stessa chiamata e risposta.


**Figura 5.15:** Primo e Terzo Ampliamento webservices, Chiamata SOAP dopo l'ampliamento

In figura 5.15 possiamo vedere la chiamata al webservice, con la richiesta di tutte le chiamate di servizio di un certo tecnico.

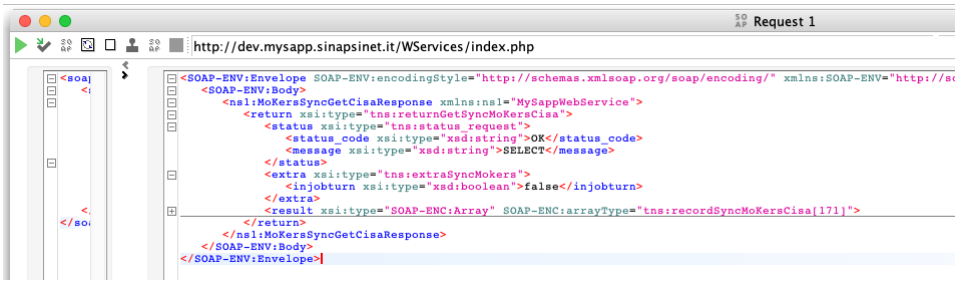



**Figura 5.16:** Primo e Terzo Ampliamento webservices, header della Risposta SOAP dopo l'ampliamento

Come possiamo vedere dallo screenshot in figura 5.16, abbiamo come messaggio di stato,
status_code, "OK", il che significa che la richiesta è andata a buon fine.


**Figura 5.17:** Primo e Terzo Ampliamento webservices, corpo della Risposta SOAP dopo l'ampliamento

In quest'altro screenshot in figura 5.17 possiamo vedere una parte interessante del lungo corpo della risposta.

In particolare possiamo vedere in una delle tante chiamate di servizio contenute nella risposta, che questa mostra nella array rappresentante la chiamata, sia **idext** che **extraj_module**, dunque gli ampliamenti sono verificati e validati.

# Capitolo 6

# Conclusioni

*In questo capitolo finale viene verificato il raggiungimento degli obiettivi prestabiliti e vengono fatte delle considerazioni sullo stato attuale del programma*

## 6.1 Raggiungimento degli obiettivi

Gli obiettivi e risultati raggiunti dall'applicazione sono stati considerati più che sufficienti dall'azienda.

In particolare sono stati raggiunti tutti gli obiettivi obbligatori concordati nel piano di lavoro.

Purtroppo non è stato possibile effettuare test automatici del codice, poichè il codice non è abbastanza corposo da renderli necessari.

I file prodotti e la stampa su finestra di messaggio sono risultati conforme alle aspettative e ne è stata verificata la correttezza.

## 6.2 Conoscenze acquisite

Le conoscenze acquisite sono state molteplici, a partire da nuovi linguaggi di programmazione al mondo dei gestionali e a tutte le applicazioni di supporto che permettono di far funzionare tutto il sistema con efficienza.

In particolare, le conoscenze acquisite principalmente sono:

* **Conoscenze sul gestionale SAP Business One:** Uno studio approfondito della documentazione SAP presente online, e uno studio parallelo su ambienti di test SAP, per testare le conoscenze via via acquisite dalla documentazione;

* **Conoscenze acquisite su applicazioni di supporto:** Molte nuove conoscenze su applicazioni come SQL Server Management Studio, Postman, SoapUI, HeidiSQL e Remote Desktop per gestire le varie parti dell'infrastruttura presistente;

* **Conoscenze acquisite su C# e librerie di SAP relative:** Uno studio della *Software Development Kit*, in breve SDK, fornita da SAP Business One, ovvero le librerie SAP per programmare gli add-ons, e l'applicazione di queste librerie su codice in C#;

* **Conoscenze acquisite su PHP:** L'aver ampliato notevolmente le conoscenze nel linguaggio PHP, dato il lavoro di ampliamento su un codice già complesso e strutturato, con necessario studio e comprensione profonda di questo codice preesistente.

## 6.3 Esperienza di stage

Nel complesso l'esperienza di studio dell'infrastruttura preesistente e sviluppo dell'add-on si è rivelata molto formativa.
Ma senza ombra di dubbio l'ampliamento delle funzioni PHP per i webservices è stata altamente formativa.

Sono soddisfatto di questo progetto, poichè sono riuscito a comprendere un po' il mondo dei gestionali e soprattutto quanta conoscenza ci sia ancora da apprendere su questo campo.
Sono soddisfatto della parte tecnologica dell'attività di stage in quanto mi ha permesso di imparare C# e approfondire il linguaggio PHP, e di applicare le mie conoscenze di SQL.

Ho potuto godere di un'ampia autonomia nella gestione ed organizzazione del mio lavoro, nel rispetto di vincoli e funzionalità stabilite ad inizio progetto.
Grazie a questa esperienza di stage ho potuto integrarmi e confrontarmi con una realtà molto diversa da quella che ero abituato a vedere e relazionarmi con persone che vedono le cose sotto un altro punto di vista.
Particolarmente importante è stata la collaborazione con i capi progetto e con i vari dipendenti dell'azienda. É stato essenziale il supporto che mi hanno fornito nell'ambientamento e nella comprensione dei nuovi argomenti, differenti da quelli studiati nel mio percorso scolastico.

# Glossario

**DBMS** In informatica, un Database Management System (abbreviato in DBMS) è un sistema software progettato per creazione, manipolazione ed interrogazione di database. Il DBMS è ospitato su architettura hardware dedicata oppure su semplice computer.. 1, 53

**UML** in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 55

**WSDL** Il Web Services Description Language (WSDL) è un linguaggio formale in formato XML utilizzato per la creazione di "documenti" per la descrizione di Web service. Il WSDL è solitamente utilizzato in combinazione con SOAP e XML Schema per rendere disponibili Web service su reti aziendali o su internet: un programma client può, infatti, "leggere" il documento WSDL relativo ad un Web service per determinare quali siano le funzioni messe a disposizione sul server e quindi utilizzare il protocollo SOAP per utilizzare una o più delle funzioni elencate dal WSDL.. 38, 40, 43, 55

# Acronimi

**AWS** Amazon Web Services. ix, 2, 3, 8, 14, 15, 23, 37, 55

**EC2** AWS Elastic Compute Cloud. 15, 55

**IDE** Integrated Development Environment. 23, 32, 55

**ODBC** Open DataBase Connectivity. 7, 8, 55

**ODBO** Object linking and embedding DataBase for Online analytical processing. 7, 55

**RDS** AWS Relational Database Service System. 15, 55

**S3** AWS Simple Storage System. 15, 55

**SAPB1** SAP Business One. 23, 55

**SOAP** Simple Object Access Protocol. x, 3, 8, 15, 17, 23, 45–49, 55

**UML** Unified Modeling Language. 30, 53

**WSDL** Web Service Description Language. 53

# Bibliografia

## Siti web consultati

*SAP B1*. URL: https://help.sap.com/viewer/product/SAP_BUSINESS_ONE/10.0/en-US/.

*SAP B1 Features*. URL: https://www.sap.com/italy/products/business-one/features.html/.

*SAP B1 Wikipedia*. URL: https://www.sap.com/italy/products/business-one/features.html/.

*SAP ERP*. URL: https://it.wikipedia.org/wiki/SAP_ERP/.

*Xamarin*. URL: https://dotnet.microsoft.com/apps/xamarin/.