


Aggiornamento Presentazione

📅 Data di inizio	@November 13, 2024
📅 Data di scadenza	@November 28, 2024
Σ Giorni rimanenti	14
# Number	2
⬇ Priority Level	 High
⚙ Status	Not started

- ☒ Analisi dei FURPS+
- ☒ Diagramma delle classi
- ☒ Diagramma dei casi d'uso
- ☒ Diagramma delle sequenze
- ☐ Architettura

▼ Analisi dei FURPS+

FURPS+ è un modello per classificare i requisiti software e include i seguenti aspetti e le relative sottocategorie:

1. Funzionalità:

- **Gestione degli asset:** Il software web app deve permettere di **aggiungere**, **modificare**, **eliminare** e visualizzare **asset IT** (PC, stampanti, periferiche..), ma anche di **licenze**, **sedi** e **dipendenti**.
- **Tracciamento e cronologia :** Il sistema deve **mantenere** uno **storico** delle **modifiche** per ogni asset.
- **Notifiche:** Il software deve inviare **notifiche** per asset che richiedono manutenzione, sono prossimi alla **scadenza** di garanzia o per le licenze prossime alla scadenza

- **Sicurezza e autenticazione:** Accesso riservato tramite autenticazione (**login con credenziali**) e gestione dei permessi (**admin e utenti base**).
 - Prossimamente: autenticazione a due fattori e/o aggiunta di passkey
- **Esportazione e importazione:** Possibilità di **esportare/importare** report degli asset in formati **CSV** o PDF.

2. Usabilità:

- **Interfaccia utente intuitiva:** L'app deve avere un'**interfaccia chiara e facile** da navigare
- **Documentazione:** Manuale utente integrato e **guida rapida all'uso** del sistema.

3. Affidabilità:

- **Disponibilità continua:** Il sistema deve essere disponibile 99,9% del tempo per garantire **accesso continuo**.
- **Backup automatici:** Creazione di **backup** regolari del database per prevenire perdita di dati.
- **Tolleranza agli errori:** Il software deve **gestire input non validi** senza crashare e fornire messaggi di errore utili.

4. Performace

- **Tempo di risposta:** ogni operazione deve avere un **tempo di risposta** molto **basso** (<1 sec)
- **Efficienza del database:** Utilizzo di MySQL ottimizzato con indici per **query veloci**.
- **Supporto a più utenti simultanei:** Il sistema deve supportare la **multi-utenza contemporanea** senza calo delle prestazioni.

5. Manutenibilità

- **Compatibilità:** **Compatibile** con i principali **browser** (Chrome, Firefox, Safari)

- Prossimamente: dispositivi mobile
- **Estensibilità:** L'app deve essere progettata per **facilitare l'aggiunta** di nuove **funzionalità** (es. integrazione con servizi di terze parti).

+ Altri requisiti

- **Design constraints:** Utilizzo di HTML, CSS e Node.js per il frontend e backend, e MySQL per il database.
- **Implementation requirements:** Scrittura del codice su Visual Studio Code e gestione del database tramite DBeaver.
- **Interface requirements:** **API** per la **comunicazione** tra **frontend** e **backend**.
- **Physical requirements:** La macchina sulla quale gira il software 8 GB di RAM e un processore dual-core per gestire il carico

▼ Diagramma delle classi

Alcune modifiche sono basate sull'incoerenza di nomi. Mi sono basato sul mio diagramma ER che è un po' più completo

Modifiche da effettuare sul diagramma delle classi

- Coerenza dei nomi
 - aggiungere consumabili
 - aggiungere componenti
 - Aggiungere sedi
- ▼ Anteprima generata da claude.ai

```
classDiagram
    class Utente {
        +id: Integer
        +nome: String
        +cognome: String
        +email: String
        +username: String
        +password: String
    }
```

```
+login()
+logout()
+visualizzaDashboard()
+validaPassword()
}

class Dashboard {
    +visualizzaGrafici()
    +passaAPremium()
    +visualizzaAzioniRecenti()
    +visualizzaSedi()
}

class Abbonamento {
    +id: Integer
    +tipo: String
    +dataInizio: Date
    +dataFine: Date
    +attiva()
    +scaduto()
}

class Pagamento {
    +id: Integer
    +importo: Float
    +data: Date
    +stato: String
    +descrizione: String
    +effettuaPagamento()
    +visualizzaDettagli()
}

class Consulenza {
    +id: Integer
    +tipo: String
    +descrizione: String
```

```

        +dataRichiesta: Date
    }

class Asset {
    <<abstract>>
    +id: Integer
    +nomeAsset: String
    +deviceImage: String
    +modello: String
    +categoria: String
    +status: String
    +dipendenteId: Integer
    +costoAcquisto: Float
    +valoreAttuale: Float
    +dataAcquisto: Date
    +add()
    +remove()
    +update()
}

class Dipendente {
    +id: Integer
    +nome: String
    +cognome: String
    +email: String
    +username: String
    +reparto: String
    +ruolo: String
    +grado: String
}

class Accessori {
    +id: Integer
    +immagine: String
    +nome: String
    +categoria: String

```

```
+modello: String
+quantitaTotale: Integer
+quantitaDisponibile: Integer
+dipendenteId: Integer
}
```

```
class Componenti {
  +id: Integer
  +nome: String
  +seriale: String
  +categoria: String
  +modello: String
  +quantitaTotale: Integer
  +quantitaDisponibile: Integer
  +dataAcquisto: Date
  +costoAcquisto: Float
}
```

```
class Consumabili {
  +id: Integer
  +nome: String
  +categoria: String
  +modello: String
  +quantitaTotale: Integer
  +quantitaRimanenti: Integer
  +numeroOrdine: String
  +dataAcquisto: Date
  +costoAcquisto: Float
}
```

```
class Licenze {
  +id: Integer
  +nomeLicenza: String
  +productKey: String
  +dataScadenza: Date
  +dataAcquisto: Date
}
```

```

+emailLegata: String
+azienda: String
+quantitaTotale: Integer
+quantitaDisponibile: Integer
}

Utente "1" -- "1" Dashboard
Utente "1" -- "*" Consulenza
Utente "1" -- "*" Abbonamento
Abbonamento "1" -- "*" Pagamento
Asset <|-- Componenti
Asset <|-- Consumabili
Asset <|-- Licenze
Dipendente "1" -- "*" Asset
Asset "1" -- "*" Accessori

```

Modifiche da effettuare sul diagramma ER

- Aggiungere tabella utenti
 - Aggiungere tipo di abbonamento e scadenza ad utente
 - Aggiungere classe abbonamento
 - Collegare utente ad abbonamento
- Aggiungere sedi
-

▼ Diagramma dei casi d'uso

Modifiche da effettuare

- Dopo richiedi consulenza (o prima) forse si potrebbe collegare ad una "fase di registrazione"
- Aggiungere un caso d'uso per la **Gestione del Profilo** dove l'utente può aggiornare i propri dati personali, cambiare la password o configurare le preferenze.

▼ Diagramma delle sequenze

Attori:

- Utente
- Browser (client)
- Server Node.js
- Database MySQL

Flusso Completo:

1. **Registrazione:** L'utente invia i dati di registrazione, che vengono salvati nel database.
2. **Login:** L'utente si autentica e riceve un token di sessione.
3. **Visualizzazione della Dashboard:** Il client invia una richiesta per caricare la dashboard, e il server risponde con i dati aggregati.
4. **Visualizzazione degli Asset:** Il client richiede la lista completa degli asset.
5. **Ricerca di Asset:** Il client invia una query di ricerca avanzata.
6. **Aggiunta di un Asset:** L'utente invia i dati di un nuovo asset.
7. **Modifica di un Asset:** L'utente modifica i dati di un asset esistente.
8. **Eliminazione di un Asset:** L'utente rimuove un asset dal database.
9. **Esportazione Dati:** L'utente richiede di esportare i dati in formato CSV.
10. **Logout:** L'utente termina la sessione.

▼ Codice .puml

```
@startuml
actor "Capo (Admin)" as Capo
actor "Ospite" as Ospite
```



```
actor Sistema
entity "Database" as DB
```

```
Capo -> Sistema: Effettua registrazione (Associa sedi)
Sistema -> DB: Salva utente, sedi, ruolo
Ospite -> Sistema: Effettua registrazione (Visualizzazione)
Sistema -> DB: Salva utente, ruolo
Capo -> Sistema: Effettua login
Ospite -> Sistema: Effettua login
Sistema -> DB: Verifica credenziali, ruolo
Sistema -> Capo: Risponde con token di sessione
Sistema -> Ospite: Risponde con token di sessione
```

```
Capo -> Sistema: Visualizza tutte le sedi
Sistema -> DB: Recupera sedi gestite dal Capo
Sistema -> Capo: Mostra sedi
Capo -> Sistema: Gestisce asset (aggiungi, modifica, elimi)
Sistema -> DB: Aggiungi/Modifica/Elimina asset (comuni a t)
Capo -> Sistema: Esporta dati (CSV)
Sistema -> DB: Recupera dati
Sistema -> Capo: Fornisce file CSV
```

```
Ospite -> Sistema: Visualizza asset
Sistema -> DB: Recupera asset (comuni a tutte le sedi)
Sistema -> Ospite: Mostra asset
Ospite -> Sistema: Logout
Capo -> Sistema: Logout
Sistema -> DB: Termina sessione
```

```
@endum1
```



▼ Architettura