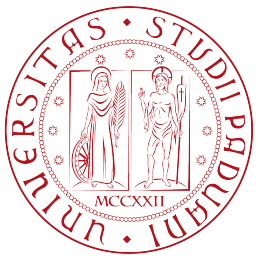

Università degli Studi di Padova

Dipartimento Matematica

Corso di Laurea Triennale in Informatica
Informatica

Uso del framework ASP.NET per la creazione di uno Schedulatore grafico delle operazioni di produzione aziendale

Relatore:
Prof. Massimiliano De Leoni

Laureando:
Canel Alessandro
1204682

Anno Accademico 2022/2023

Indice

1 Introduzione **1**
- 1.1 Introduzione . . . 1
  - 1.1.1 Scopo del documento . . . 1
  - 1.1.2 Convenzioni tipografiche . . . 1
  - 1.1.3 Struttura del documento . . . 2

2 Lo Stage **3**
- 2.1 L'Azienda . . . 3
- 2.2 Informazioni Azienda . . . 3
- 2.3 Scopo dello Stage . . . 4
- 2.4 Analisi preventiva dei rischi . . . 6
- 2.5 Risultati attesi . . . 6
- 2.6 Contenuti formativi previsti . . . 6
- 2.7 Pianificazione del Lavoro . . . 7

3 Analisi dei requisiti **9**
- 3.1 Specifica casi d'uso . . . 9
- 3.2 Tracciamento dei requisiti . . . 14

4 Progettazione e Codifica **17**
- 4.1 Tecnologie . . . 17
  - 4.1.1 Il sistema operativo . . . 17
  - 4.1.2 Browser web . . . 17
  - 4.1.3 IDE^G e controllo versione^G . . . 18
  - 4.1.4 Piattaforme a scopo collaborativo . . . 18
  - 4.1.5 Librerie usate . . . 19
- 4.2 Architettura ASP.NET . . . 20
  - 4.2.1 Javascript nel contesto .NET . . . 21
  - 4.2.2 Modello MVC . . . 21
- 4.3 Organizzazione del Lavoro . . . 21
- 4.4 Bryntum . . . 23
  - 4.4.1 Progettazione . . . 23
  - 4.4.2 Codifica . . . 26
  - 4.4.3 Pagina principale . . . 29
  - 4.4.4 Chiamate alla libreria . . . 31

4.4.5 Problematiche rilevate 32
4.5 DevExpress 33
4.5.1 Progettazione 33
4.5.2 Codifica 35
4.6 NetTronic 38
4.7 Design pattern 39
4.8 Project menagement 39

5 Conclusioni **41**
5.1 Generale 41
5.2 Scelta libreria 41
5.3 Rispetto requisiti funzionali 43
5.3.1 Bryntum 43
5.3.2 DevExpress 45
5.4 Valutazioni personali 47
5.5 Termine stage e possibilità future 47
5.6 Conoscenze Acquisite 48

6 Glossario **49**

7 Fonti **53**

Elenco delle tabelle

3.1 Tabella del tracciamento dei requisti funzionali 15
3.2 Tabella del tracciamento dei requisti qualitativi 16

Elenco delle figure

2.1 Logo azienda dal sito 3
2.2 Esempio Istogramma **RF[DES]8** vecchio prodotto 5

3.1 Diagramma casi d'uso 9

4.1 Logo Citrix 17
4.2 Piattaforme di instant sharing 18
4.3 Esempio di organizzazione in un progetto ASP.NET 20
4.4 Schema pattern Model View Controller 22
4.5 Mockup pagina web 23
4.6 Classi d'interesse per lo sviluppo 25
4.7 Primo prodotto Bryntum con gantt 26
4.8 Esempio feature abbandonata passando da diagramma gant a scheduler 26
4.9 Interfaccia Bryntum con una attività divisa in due 27
4.10 Dipendenze mantenute 28
4.11 Pulsanti zoom 29
4.14 Istogramma con massimo numero di ore 29
4.12 Implementazione dello scheduler con l'istogramma 30
4.13 Implementazione dello scheduler, istogramma e gantt in contemporanea 30
4.15 Pulsante utilizzato per dividere una attività dopo la selezione . . . 30
4.16 Sequence diagram delle prime chiamate necessarie 31
4.17 Mockup Devexpress 34
4.18 Pulsanti per cambio delle viste 35
4.19 Struttura classi per le viste DevExpress 36
4.20 Scheduler DevExpress Settimana 36
4.21 Scheduler DevExpress Mese 37
4.22 Scheduler DevExpress Mese prima delle modifiche 37
4.23 Un esempio di un possibile prodotto NetTronic 38

Capitolo 1

Introduzione

1.1 Introduzione

1.1.1 Scopo del documento

Il seguente documento ha lo scopo di esporre l'attività di stage da parte del laureando Canel Alessandro presso l'azienda Eurosystem, dal 02/05/2022 al 22/06/2022 per un totale di 300 ore.
Lo stage, da riferirsi alla sede Eurosystem SpA di Villorba (TV), è stato tenuto in modalità mista in presenza o da remoto con l'impiego degli strumenti di condivisione remota forniti dall'azienda (Microsoft Teams, Citrix, accessi in vpn ai server aziendali necessari) assicurando un confronto diretto con il tutore aziendale per ogni necessità. L'orario di lavoro è avvenuto dal lunedì al venerdì dalle 9.00 alle 13.00 e dalle 14.00 alle 18.00.

Lo stage si è suddiviso in attività settimanali che a loro volta possono essere suddivise in tre parti

- **Studio della libreria**: Ogni settimana veniva approfondita una libreria necessaria per lo sviluppo di un un Proof of Concept^G di un prodotto in linea con le indicazioni del relatore di azienda.
- **Implementazione**: La fase successiva, cioè dove le conoscenze apprese dalla ricerca nella prima fase possono essere applicate a scopo di creare un prodotto valutabile.
- **Valutazione**: Lo scopo della revisione settimanale era di valutare i pro e i contro di ogni libreria e se essa soddisfa i requisiti minimi.

Ogni settimana veniva creata un Proof of Concept^G della libreria a scopo d'illustrarne le features^G e stillare una lista dei requisiti minimi soddisfatti.

1.1.2 Convenzioni tipografiche

Al fine di ottimizzare la comprensione del documento, verranno utilizzate le seguenti convenzioni tipografiche:

- i termini specifici verranno identificati da una "G" in apice e il loro significato verrà descritto nella sezione Glossario, consultabile alla fine del documento
- le parole o le frasi di maggior rilevanza verranno evidenziate in **grassetto**

1.1.3 Struttura del documento

La tesi è suddivisa nei seguenti capitoli:

- **Introduzione**: Introduce lo scopo del documento e dello stage
- **Lo Stage**: In questo capitolo verrà introdotta l'azienda, il progetto dello stage, i requisiti principali e secondari del prodotto da sviluppare e la pianificazione del lavoro.
- **Analisi dei requisiti**: In questo capitolo verranno descritti i requisiti funzionali e qualitativi del prodotto.
- **Progettazione e codifica**: Progettazione e codifica dei vari prodotti creati in azienda.
- **Glossario**: Il glossario viene utilizzato per rendere di più facile lettura il documento. Le definizioni sono tratte da articoli presenti nelle fonti.
- **Fonti**: In questo capitolo vengono elencate le fonti utilizzate per la creazione di questo documento e le librerie studiate durante il percorso di stage.

Capitolo 2

Lo Stage

2.1 L'Azienda

Eurosystem S.p.A. è una nota società nel campo dell'Information and Communication Technology. La sua storia inizia nel 1979 a Treviso e continua all'interno di un percorso evolutivo costantemente in crescita. Si presenta sul mercato del Nord e Centro Italia con cinque sedi (Treviso, Bergamo, Bologna, Udine, Ferrara), 150 professionisti interni e oltre 1300 clienti. A oggi è diventata un gruppo di più realtà specializzate che mira a garantire soluzioni informatiche di qualità per la trasformazione digitale e l'Industria 4.0^{G}.

2.2 Informazioni Azienda

- RAGIONE SOCIALE: EUROSYSTEM SPA
- REFERENTE AZIENDALE: LISA FRANZOSO
- INDIRIZZO - Via Newton, 21 - Villorba (TV)
- TELEFONO - 3487226003
- SITO WEB - hiips://www.eurosystem.it/
- EMAIL - hr@eurosystem.it

Figura 2.1: Logo azienda dal sito

2.3 Scopo dello Stage

Il progetto riguarda la creazione di uno schedulatore^G grafico delle operazioni di produzione. Nell'ambito della manifattura di prodotti finiti o semilavorati, un software ERP (Enterprise resource planning) è dotato dello strumento FCS (Finite Capacity Scheduling) che organizza le operazioni di produzione al dettaglio nei centri di lavoro.
All'interno dell'azienda vengono utilizzati anche strumenti MPS (Master Production Schedule) e MRP (Material Requirements Planning) che affrontano gli stadi di pianificazione per famiglie o aggregazioni di articoli, ma non sono d'interesse per lo scopo di stage. Nell'ottica del rifacimento del software FCS di Eurosystem in tecnologia web, lo stage intende esplorare una o più librerie di scheduler grafici per il web realizzando prototipi dimostrativi in tecnologia ASP.NET che saggino le caratteristiche grafiche interattive adatte alla realizzazione di un nuovo software FCS.
Quindi lo stage è mirato alla creazione di una applicazione web utilizzata proprio per organizzare e visualizzare le operazioni di produzione dell'azienda. Il prodotto deve essere creato mediante ASP.NET, con lo scopo di ottenere un'interfaccia web responsive^G, allo scopo poter organizzare le varie attività di produzione fornite dalla azienda nei vari centri di lavoro.
La valutazione delle librerie durante il percorso di stage è alla possibilità di trovare una soluzione in grado di sostituire un prodotto creato inizialmente nei primi anni 2000 da parte dell'azienda.
Il prodotto offriva una rappresentazione Gantt^G e Istografica delle attività. Il prodotto era mirato all'utilizzo in aziende edili o cantieri, e aveva lo scopo di creare organigrammi.
All'inizio dell'esperienza di stage mi è stato proposto di guardare un video di un addetto allo sviluppo software che spiegava in dettaglio il funzionamento del prodotto da sostituire (FCS). La immagine 2.2 è mostrata una schermata rappresentante le funzionalità principale del prodotto. Tali funzionalità verranno poi descritte più a fondo nei capitoli successivi.

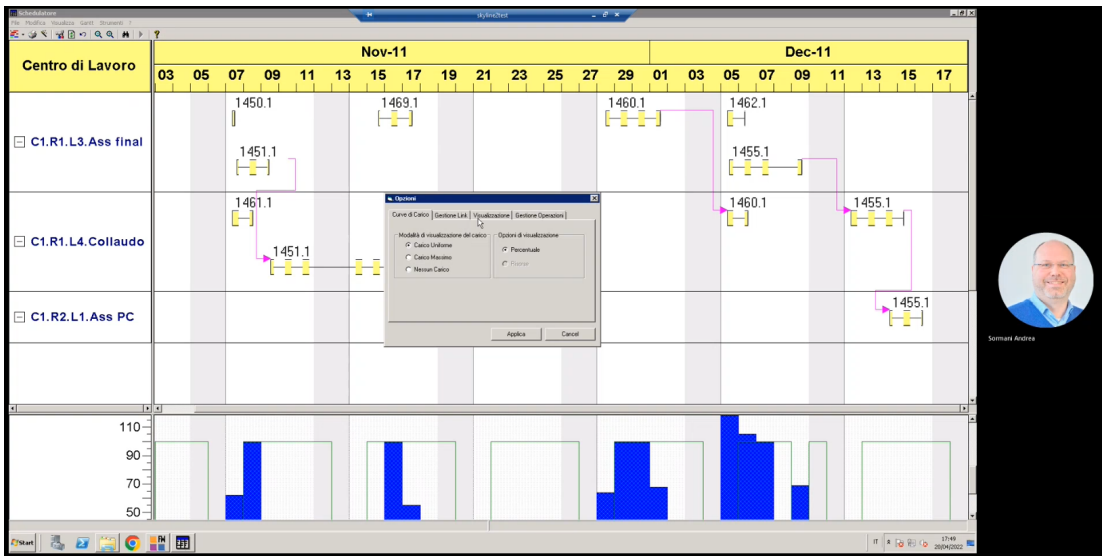

Figura 2.2: Esempio Istogramma **RF[DES]8** vecchio prodotto

2.4 Analisi preventiva dei rischi

I rischi legati allo stage sono relativi alle tecnologie non ancora conosciute dallo studente. Per far fronte a questa problematica ho pianificato le due settimane iniziali di acquisizione conoscenze di base sugli strumenti da impiegare.
Queste settimane consistono in una stretta comunicazione tra il tutore aziendale e lo studente, a scopo di facilitare il processo formativo.
Altri rischi potrebbero subentrale durante la stesura del codice nelle macchine fornite dalla azienda, a causa della novità dell'hardware proposto (interfacciarsi mediante RasperryPI^G a una macchina virtuale).

2.5 Risultati attesi

Al termine dello stage lo studente avrà acquisito conoscenze di programmazione in ambito ASP.NET e web client utili alla realizzazione di applicazioni di front-end^G che utilizzano librerie grafiche di pianificazione (ad esempio Gantt, istogrammi ecc); inoltre lo studente avrà una visione generale a riguardo delle attività di pianificazione della produzione e dei ruoli aziendali coinvolti.
Al termine del periodo di stage, grazie alla esperienza in azienda, produrrà un elaborato che verrà utilizzato per la Tesi di Laurea.

2.6 Contenuti formativi previsti

Durante la durata dello stage lo studente approfondirà le sue conoscenze in materia dei processi aziendali inerenti la pianificazione della produzione e user interfaces adatte allo svolgimento di attività in tale ambito. Il piano di stage prevede che lo studente sia coinvolto in:

- Impiego della tecnologia ASP.NET e Microsoft Visual Studio.NET
- Sviluppo soluzioni web di front-end^G
- Impiego di librerie grafiche adatte alla pianificazione della produzione (ad es. Gantt e istogrammi)

2.7 Pianificazione del Lavoro

- **Prima e Seconda settimana** - 02/05/2022 - 06/05/2022 e 09/05/2022 - 13/05/2022 (80 ore)
  - Incontro con il reparto sviluppo software e discussione requisiti e informazione sui prototipi da sviluppare;
  - Formazione su tecnologie adottate e librerie grafiche da analizzare;
  - Formazione su dispositivi forniti (PC aziendale);
  - Formazione su infrastruttura esistente;
  - Introduzione generale alla pianificazione della produzione.
- **Terza e Quarta settimana** - 16/05/2022 - 20/05/2022 e 23/05/2022 - 27/05/2022 (80 ore)
  - Continuazione della formazione;
  - Analisi casi d'uso;
  - Realizzazione di prototipi esplorativi impieganti le librerie grafiche selezionate, per esplorarne la capacità a supporto dei requisiti dei casi d'uso (selezione della libreria più adeguata);
  - Stesura della architettura e specifiche progetto;
  - Stesura documentazione.
- **Quinta e Sesta settimana** - 30/05/2022 - 03/06/2022 e 6/06/2022 - 10/06/2022 (80 ore)
  - Continuazione della documentazione;
  - Stesura codice secondo analisi;
  - Analisi e studio approfondito delle tecnologie specifiche della libreria grafica scelta;
  - Inizio prodotto finale;
- **Settima e Ottava settimana** - 13/05/2022 - 17/06/2022 e 20/06/2022 - 22/06/2022 (60 ore)
  - Continuazione sviluppo prodotto;
  - Termine Collaudo del prodotto finale;
  - Esposizione prodotto terminato.

Capitolo 3

Analisi dei requisiti

3.1 Specifica casi d'uso

Nella seguente immagine viene illustrato un grafico con i casi d'uso schematizzati tramite UML^G. Ogni caso d'uso rappresenta una parte del software con cui l'utente (unico soggetto) interagisce. I casi d'uso sono poi associati a dei requisiti funzionali, elencati nella tabella sotto.
Esiste solo un attore primario:
L'utente: Cioè colui che deve interagire con il grafico per aggiungere/rimuovere/modificare le attività.

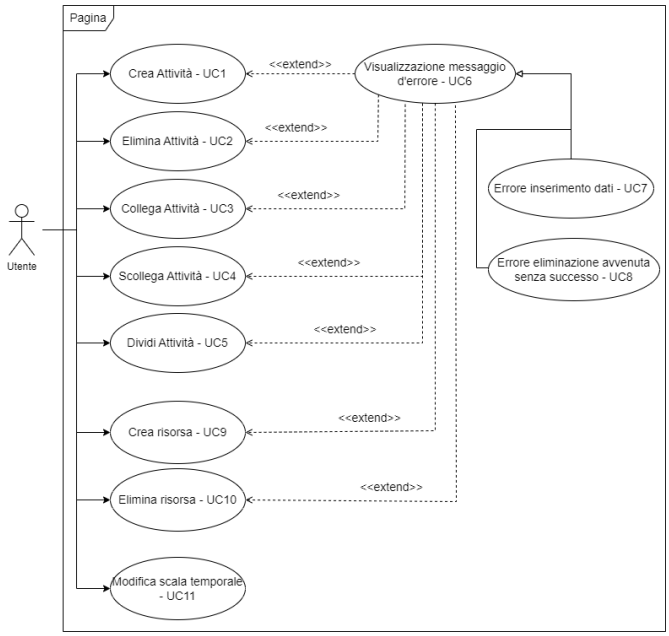

Figura 3.1: Diagramma casi d'uso

UC1: Crea attività

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale.
Descrizione: l'utente inserisce l'attività con un tocco sulla riga della risorsa a cui vuole assegnare l'attività.
Scenario Principale:

- l'utente seleziona il luogo nel diagramma dove desidera che venga inserita l'attività
- l'utente modifica i vari dati inerenti alla attività in creazione
- l'utente conferma i dati inseriti tramite l'apposito pulsate

La libreria controlla la correttezza dei dati in termini logici. **Postcondizioni**: l'utente ha inserito con successo l'attività e una barra rappresentante quest'ultima appare nel diagramma. Le ore vengono inoltre aggiunte all'istogramma sotto.

UC2: Elimina attività

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale.
Descrizione: l'utente elimina l'attività che desidera eliminare.
Scenario Principale:

- l'utente seleziona l'attività
- l'utente seleziona "elimina attività" dal menù a tendina
- l'utente conferma la eliminazione

Postcondizioni: l'utente ha eliminato con successo l'attività e l'icona rappresentante l'attività. Le ore vengono poi eliminate dall'istogramma sotto.

UC3: Collega attività

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale
Descrizione: l'utente collega due attività con una relazione.
Scenario Principale:

- l'utente seleziona l'attività
- l'utente trascina con il puntatore da una estremo di una attività all'estremo dell'altra
- l'utente conferma il collegamento

Postcondizioni: l'utente ha collegato con successo le attività e nel diagramma compare una freccia rappresentante tale relazione.

UC4: Scollega attività

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale
Descrizione: l'utente elimina la relazione tra due attività.
Scenario Principale:

- l'utente seleziona il collegamento
- l'utente seleziona "elimina attività" dal menù a tendina
- l'utente elimina il collegamento

Postcondizioni: l'utente ha eliminato con successo le attività e nel diagramma la freccia rappresentante tale relazione viene rimossa.

UC5: Dividi attività

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale.
Descrizione: l'utente divide una attività.
Scenario Principale:

- l'utente seleziona l'attività
- l'utente seleziona "Dividi" nella interfaccia sopra
- l'utente conferma la divisione

Postcondizioni: l'utente ha diviso con successo l'attività e l'icona rappresentante l'attività. Le ore vengono poi sommate nell'istogramma sotto.

UC6: Visualizzazione messaggio d'errore

Attori Principali: utente.
Precondizioni: l'utente ha inserito dei dati non corretti.
Descrizione: l'utente inserisce dei dati non corretti quindi viene quindi visualizzato un messaggio d'errore.
Specializzazioni:

- Visualizzazione errore inserimento dati - **UC7**
- Visualizzazione eliminazione avvenuta senza successo - **UC8**

Postcondizioni: viene visualizzato il messaggio di errore apposito per il tipo di errore

UC7: Visualizzazione errore inserimento dati

Attori Principali: utente.
Precondizioni: l'utente ha inserito dei dati non corretti o mancanti.
Descrizione: l'utente inserisce dei dati non corretti quindi viene quindi visualizzato un messaggio d'errore.
Scenario Principale: l'utente inserisce dei dati errati, su una risorsa inesistente o con delle date non logicamente possibili.
Postcondizioni: viene visualizzato il messaggio di errore apposito.

UC8: Errore eliminazione avvenuta senza successo

Attori Principali: utente.
Precondizioni: l'utente ha tentato di eliminare una attività.
Descrizione: l'utente ha tentato di eliminare la entità ma la eliminazione non è andata a buon fine.
Scenario Principale: l'utente non elimina la entità.
Postcondizioni: viene visualizzato il messaggio di errore apposito.

UC9: Crea risorsa

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale.
Descrizione: l'utente crea una nuova risorsa.
Scenario Principale:

- l'utente seleziona il pulsante inerente alla creazione risorsa
- l'utente modifica i vari dati inerenti alla attività in creazione
- l'utente conferma i dati inseriti tramite l'apposito pulsante

La libreria controlla la correttezza dei dati in termini logici.
Postcondizioni: l'utente ha inserito con successo la risorsa e compare a fianco di entrambi i diagrammi.

UC10: Elimina risorsa

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale
Descrizione: l'utente elimina la risorsa che desidera eliminare.
Scenario Principale

- l'utente seleziona la risorsa
- l'utente seleziona "elimina risorsa" dal menù a tendina

- l'utente conferma l'eliminazione

Postcondizioni: l'utente ha eliminato con successo la risorsa e l'icona rappresentante.

UC11: Modifica scala tempo

Attori Principali: utente.
Precondizioni: l'utente si trova nella pagina principale
Descrizione: l'utente seleziona le icone dedicate alla modifica della scala temporale.
Scenario Principale

- l'utente seleziona il tasto
- l'asse temporale si comporta di conseguenza

Postcondizioni: l'utente ha modificato con successo l'asse temporale.

3.2 Tracciamento dei requisiti

In questo paragrafo verranno tabulati i requisiti funzionali e qualitativi.
Ogni requisito viene accompagnato da una descrizione e la fonte. In seguito viene descritto come i codici di ogni requisito vengono composti per aiutarne la comprensione.
La tipologia del codice è la seguente: **Tipologia[Priorità](X.Y)**, Dove:

- **R**: Requisito
- **F**: Funzionale
- **Q**: Qualitativo
- **Tipologia**: indica la tipologia del requisito, che può essere
- - **OBB**: obbligatoria
  - **DES**: desiderabile
  - **OPZ**: opzionale
- **X, Y**: Indica la numerazione per ordinare i requisiti in maniera gerarchica

Codice	Descrizione	Fonti
RF[OBB]1	Il prodotto deve essere utilizzabile da interfaccia web	Committente
RF[OBB]2	Il prodotto deve poter creare ed eliminare delle attività per ogni centro lavorativo	UC1
RF[OBB]3	Bisogna essere in grado di creare o rimuovere enti	UC1/UC2
RF[OBB]4	Deve essere presente un istogramma per mostrare a colpo d'occhio le ore lavorative per ciascun individuo	Committente
RF[OBB]5	Le attività devono essere collegabili tra di loro	UC3
RF[OBB]6	Le attività devono essere divisibili	UC5
RF[OBB]6.1	Tale divisione deve portare a una divisione delle previe dipendenze, lasciando l'attività virtualmente uguale in termini di flusso lavorativo	UC5
RF[OBB]6.2	Eliminare una attività deve eliminare anche le dipendenze o dare avviso della loro imminente eliminazione	UC2
RF[OBB]7	Deve essere tutto contenuto in una istanza web, senza utilizzo di file temporanei (come ad esempio cookies)	Committente
RF[OBB]8	Deve esistere una gerarchia stabilimento/ reparto/centro di lavoro	UC9
RF[OBB]8.1	Le gerarchie devono essere flessibili al punto di poter trasferire una attività da un centro di lavoro all'altro, o suddividere una attività per operazioni in parallelo di vari centri di lavoro	UC9
RF[OBB]9	Deve essere possibile interagire con la risoluzione della linea temporale, come visualizzare in mesi, settimane o giorni	Committente
RF[DES]10	Le attività raggruppate possono essere collassate per una visione più agevole	Committente
RF[DES]11	L'istogramma delle ore di lavoro mostra quando le ore assegnate sono superiori alle ore assegnabili	Committente
RF[DES]12	La funzione di divisione delle attività deve suddividere anche a livello temporale, cosicché una attività si suddivida in due mezze attività	UC5
RF[FAC]13	Mostrare la percentuale di attività già compiuta	Committente

Tabella 3.1: Tabella del tracciamento dei requisti funzionali

Codice	Descrizione	Fonti
QF[OBB]1	Il codice deve essere condiviso sulla cartella pubblica	Committente
QF[OBB]2	La pagina web deve essere responsive^G	Committente
QF[OBB]3	La pagina web deve essere intuitiva	Committente
QF[FAC]4	La pagina web deve essere mobile-friendly^G	Committente

Tabella 3.2: Tabella del tracciamento dei requisti qualitativi

Capitolo 4

Progettazione e Codifica

4.1 Tecnologie

In questa sezione verranno elencati i sistemi operativi e software installati utilizzati dall'azienda. L'hardware fornito dall'azienda è RaspberryPI^G dotato di tastiera e mouse collegato a un monitor.

4.1.1 Il sistema operativo

I computer presenti in azienda e quello provveduto per il lavoro in remoto hanno il sistema operativo Citrix preinstallato. Citrix ha lo scopo d'interfacciarsi con una macchina virtuale Windows remota. L'accesso avviene fornendo le proprie credenziali di login date dall'azienda e una password creata all'inizio dell'apprendistato. Al termine dell'esperienza in azienda le informazioni di login vengono eliminate, assieme ai dati presenti all'interno della VM.

Figura 4.1: Logo Citrix

4.1.2 Browser web

I browser preinstallati sono tre: **Mozilla Firefox**, **Google Chrome** e **Microsoft Edge**. Nella fase di sviluppo ho scelto di utilizzare Edge dato che ero già familiare con il browser. In più ho trovato gli strumenti per sviluppatori già integrati in Edge molto completi e indispensabili per analizzare a fondo le librerie e il loro funzionamento.

4.1.3 IDE^G e controllo versione^G

In questo paragrafo descrivo gli strumenti utilizzati per scrivere codice e condividerlo con i tutori.

- **Visual Studio 2019**^{G}: Visual Studio è preinstallato su ogni macchina e funge da IDE^G principale per ogni postazione di lavoro, ed è l'editor di testo su cui i tutori d'azienda erano più preparati.
- **Controllo versione**^{G}: Ogni progetto veniva caricato iterativamente da versione a versione in una cartella nella rete aziendale. Ogni utente dell'azienda aveva una cartella personale, dove caricavo personalmente i miei progressi da settimana a settimana.

4.1.4 Piattaforme a scopo collaborativo

Le seguenti piattaforme sono state utilizzate durante il percorso di stage per agevolare la collaborazione tra i diversi membri dell'azienda.

- **Microsoft Teams** è una piattaforma d'instant messaging sviluppata da Microsoft. Nel mia esperienza in azienda ho utilizzato Teams per collaborare assieme ai miei tutori anche da remoto.
  Mi ha inoltre permesso di esporre delle domande senza organizzare meeting in presenza.
- **Outlook** è invece stato utilizzato per l'invio e ricevimento di email e per la sua opzione di calendarizzazione degli eventi. Ogni qualvolta un utente ha bisogno di organizzare un meeting in una stanza specifica può fissarla come evento all'interno del calendario. L'evento creato viene sincronizzato con tutti i calendari dell'azienda per evitare conflitti con altri membri.
  Una volta inviata la richiesta si riceve una mail di conferma della prenotazione della stanza. Una tale prenotazione è avvenuta per il meeting con il tutore universitario durante i due incontri in chiamata Zoom, dove ho prenotato una stanza per non disturbare terze parti.
- **Zoom** è stata una piattaforma utilizzata solo per le chiamate con il tutore universitario e durante l'ultima settimana per il lavoro da remoto.

Figura 4.2: Piattaforme di instant sharing

4.1.5 Librerie usate

- **jQuery**^{G} è una libreria^G utilizzata per interagire con il DOM^G di una pagina web, le prime due settimane si sono concentrate principalmente nello svolgimento di progetti di piccola scala per consentirmi d'imparare la tecnologia. I progetti forniti dal tutore a scopo didattico erano delle piccole implementazioni della libreria, dove il sottoscritto doveva rispettare i vincoli imposti dal tutore.

- **Node.js**^{G} è un ambiente di runtime JavaScript open-source^G, multipiattaforma che esegue codice JavaScript al di fuori di un browser Web.

- **Bryntum**^{G} è una libreria che ha spiccato l'interesse dei tutori aziendali per la possibilità di collassare task all'interno del diagramma di Gantt^G.

- **DevExpress/DevExtreme**^{G} è libreria che ha attirato l'Interesse dei tutori di azienda per la gran quantità di grafici a disposizione e dal fatto che membri dell'azienda avevano già esperienza con essa.

4.2 Architettura ASP.NET

.NET è una piattaforma di sviluppo costituita da strumenti, linguaggi di programmazione e librerie per la creazione di molti tipi diversi di applicazioni.
La maggior parte delle applicazioni .NET tradizionali vengono costruite come singole unità o riducibili singola applicazione Web in esecuzione all'interno di un singolo dominio^G.
Questo approccio è il più semplice modello di distribuzione ed è il modello utilizzato da molte applicazioni web. La maggior parte delle applicazioni aziendali più complesse necessitano di una minima separazione, detta a "strati" o layered. Essendo in sviluppo un prodotto semplice e a puro scopo dimostrativo, si è optato per una architettura detta monolitica.
Un'applicazione monolitica è un'applicazione completamente autonoma, in termini di comportamento. Potrebbe interagire con altri servizi o database nel corso dell'esecuzione delle sue operazioni, ma il nucleo del suo comportamento viene eseguito all'interno del proprio processo e l'intera applicazione viene generalmente distribuita come una singola unità. Se un'applicazione di questo tipo deve essere ridimensionata orizzontalmente, in genere l'intera applicazione viene duplicata più server o macchine virtuali.
Un nuovo progetto ASP.NET, creato in Visual Studio o dalla riga di comando, inizia come un semplice monolite "all-in-one". La Figura 4.3 mostra la suddivisione in cartelle di un tale progetto, fornito dal sito ufficiale della piattaforma .NET.

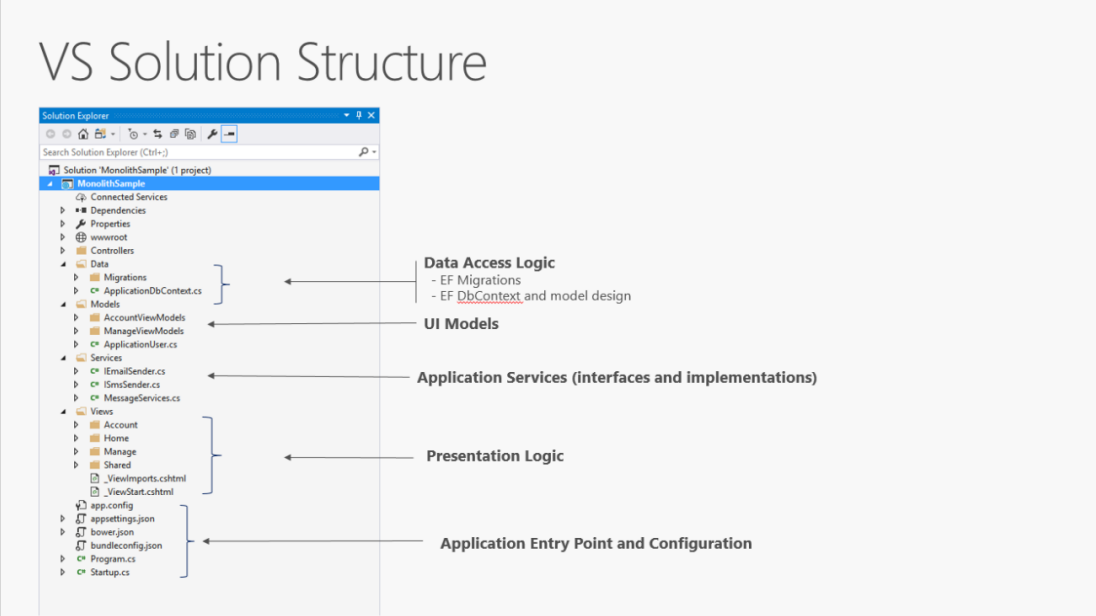

Figura 4.3: Esempio di organizzazione in un progetto ASP.NET

In un progetto sinlavoragolo come quello sviluppato, la separazione tra files avviene mediante l'uso di cartelle. Il modello di organizzazione è chiamato modello MVC ed è composto da modelli, viste e controller.

In questa disposizione, i dettagli della presentazione vista dall'utente dovrebbero essere limitate il più possibile alla cartella **Views**.
I dettagli di implementazione dell'accesso ai dati dovrebbero essere limitate alle classi conservate nella cartella **Data**.
La logica dovrebbe risiedere nei servizi e nelle classi all'interno la cartella **Model**.
Sebbene semplice, la soluzione monolitica a progetto singolo presenta alcuni svantaggi. Con l'aumentare della dimensione del progetto la complessità aumenta, e con quello anche il numero di file e cartelle continuerà a crescere.
I files (modelli, viste, controller) risiedono in più cartelle, che non sono raggruppate insieme in ordine alfabetico e quindi sono disorganizzate.

4.2.1 Javascript nel contesto .NET

JavaScript è un linguaggio di programmazione dinamico^G e interpretato^G. È il linguaggio di programmazione del web. Come CSS, JavaScript può essere implementato all'interno di elementi HTML, come blocchi di script all'interno di una pagina o in files separati. Quando si lavora con JavaScript in una applicazione web, in genere si:

- Seleziona di un elemento HTML e si eseguono modifiche.
- Utilizza un'API Web.
- Invia un comando a un'API Web (e risposta a una richiamata con il relativo risultato).

Una delle librerie più popolari è jQuery, che continua a essere una scelta popolare per semplificare queste attività. Per le applicazioni a pagina singola, jQuery fornisce molte delle funzionalità desiderate offerte da Angular e React, infatti questi prodotti sono stato realizzati implementando jQuery.

4.2.2 Modello MVC

ASP.NET MVC è un framework^G basato sul pattern Model View Controller.
Il **Controller** risponde all'input dell'utente, e si occupa del comportamento dell'applicazione, infatti elabora gli input in arrivo, aggiorna il **Model** corrispondente e fornisce la **View** pertinente. Ogni richiesta o azione è mappata da un controller. L'MVC fornisce il risultato di una serie di chiamate di metodo (vedi immagine 4.4). La **View** invia l'aggiornamento al **Controller**, il **Controller** aggiorna il **Model** e il **Model** aggiorna direttamente la **View**.

4.3 Organizzazione del Lavoro

Come citato nell'introduzione, lo scopo di questa esperienza era d'implementare delle librerie a scopo d'informare l'azienda nella scelta di un prodotto che soddisfi

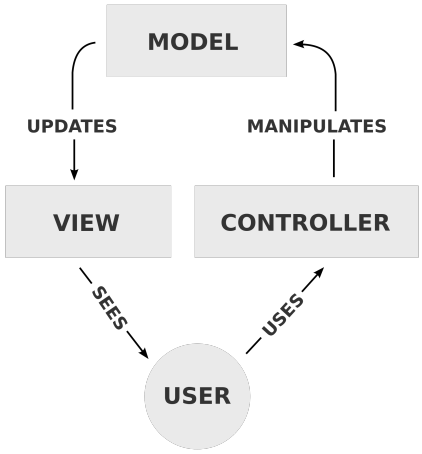

Figura 4.4: Schema pattern Model View Controller

il più gran numero di requisiti imposti.
Lo sviluppo è avvenuto quindi tramite un ciclo settimanale dove la codifica e la progettazione avvenivano contemporaneamente e ogni fine settimana avveniva una analisi da parte dei tutori, nella quale esprimevano il proprio giudizio sul lavoro svolto.
I meeting settimanali mi permettevano inoltre di ottenere degli aggiustamenti da parte del committente (che in questo particolare caso coincideva con il tutore aziendale) in tempo reale.

4.4 Bryntum

4.4.1 Progettazione

Le parti che durante le prime analisi sono risultate più onerose sono:

- L'implementazione della funzione **split()**
- L'avanzamento percentuale delle attività
- La divisione tra diagramma Gantt e Scheduler
- Creazione e mantenimento delle dipendenze

Una volta creato il mockup^G della pagina principale ispirandosi ai prodotti nel sito della libreria è nata una primissima implementazione mostrata in figura 4.5.

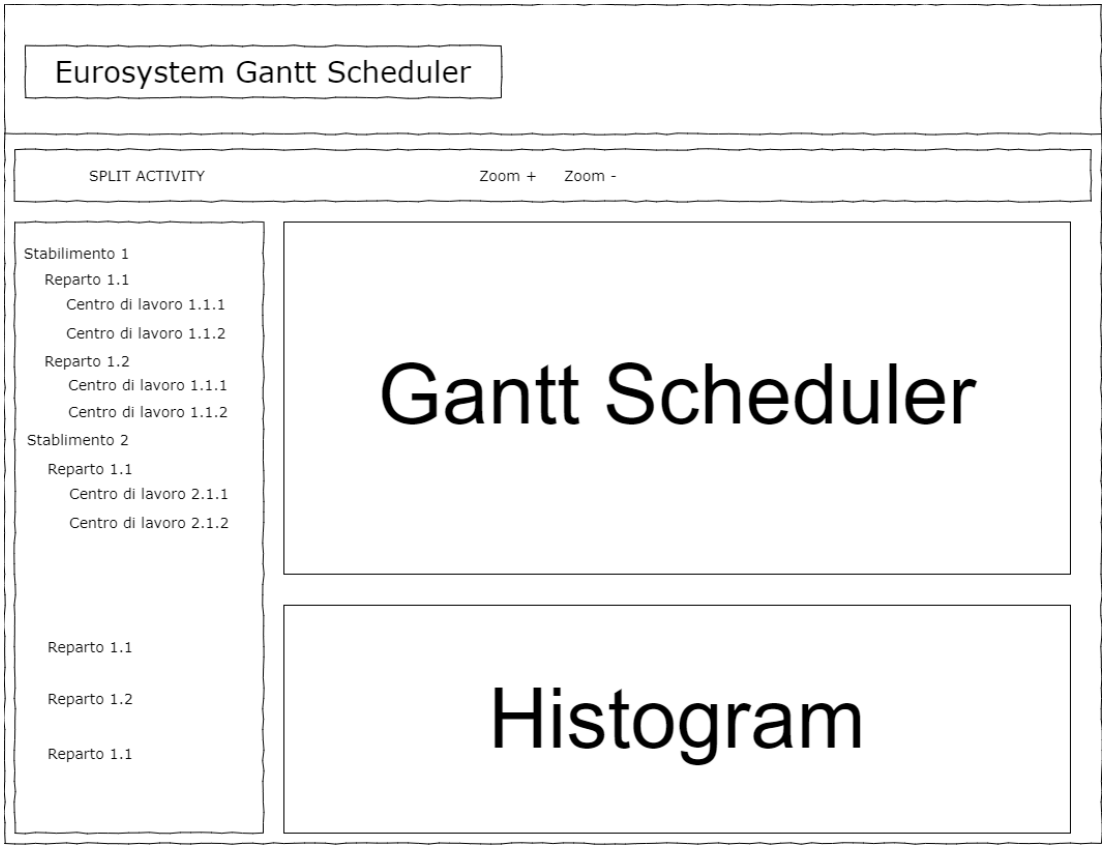

Figura 4.5: Mockup pagina web

Studio organizzazione classi

Uno **Store** rappresenta un contenitore con svariati tipi di tipologie di dati. Un articolo nello Store viene spesso chiamato un "record" ed è semplicemente un'istanza del Modello (o di qualsiasi sua sottoclasse).
Un **ResourceStore** infatti utilizza un **ResourceModel** e un **EventStore** utilizza un **EventModel**.
Entrambi ereditano i metodi da **Model**, ma differiscono nei tipi di campi dati che contengono. Bryntum Scheduler utilizza **ResourceStore** per tenere le risorse e un **EventStore** per gli eventi.
Una risorsa è ad esempio un centro di lavoro e un evento è ad esempio la produzione di chiodi. Entrambi sono inclusi in una classe base **Store**, che contiene e gestisce i dati in formato JSON^G.
Lo Scheduler utilizza anche **Store** aggiuntivi, come **AssignementStore** e **DependencyStore**.

- **AssignementStore**: Uno **Store** che rappresenta un gruppo di assegnazioni tra gli eventi nell'**EventStore** e le risorse nel **ResourceStore** (ad esempio questo lega la postazione di lavoro con la produzione di dadi). Un **AssignementStore** è in genere connesso a un elemento project, che lo lega ad altri archivi correlati (EventStore, ResourceStore e DependencyStore). Il progetto gestisce anche i riferimenti (evento, risorsa) correlati nello **Store**.
  La risoluzione di possibili modifiche avviene in modo asincrono, non è garantito che i record abbiano riferimenti aggiornati fino al termine delle elaborazioni (un esempio potrebbe essere la modifica di una attività che non viene mostrata fino al nuovo render). Per essere certi che i riferimenti vengano risolti, si chiama **await project.commitAsync()**.

- **DependencyStore**: Uno **Store** che rappresenta un gruppo di dipendenze tra eventi in **EventStore** (ad esempio questo lega la produzione di chiodi con la produzione di dadi).
  Un DependencyStore è in genere connesso a un elemento project, che lo lega ad altri archivi correlati (EventStore, AssignmentStore e ResourceStore). Il progetto gestisce anche i riferimenti (fromEvent, toEvent) correlati nello **Store**. La risoluzione dei riferimenti avviene in modo asincrono, non è garantito che i record abbiano riferimenti aggiornati fino al termine dei calcoli. Per essere certi che i riferimenti vengano risolti, chiama **await project.commitAsync()**.

La funzione **await project.commitAsync()** è necessaria per assicurarsi che tutte le modifiche avvenute durante il processo dei dati per una funzione vengano eseguite e salvate.
Gli **store** degli scheduler sono collegati tra loro tramite un **project**. Un **project** può essere pensato come il set di dati completo a disposizione dello Scheduler: tutti gli eventi, le risorse, le assegnazioni e le dipendenze sotto un unico "parent" (il project) come da immagine 4.6.

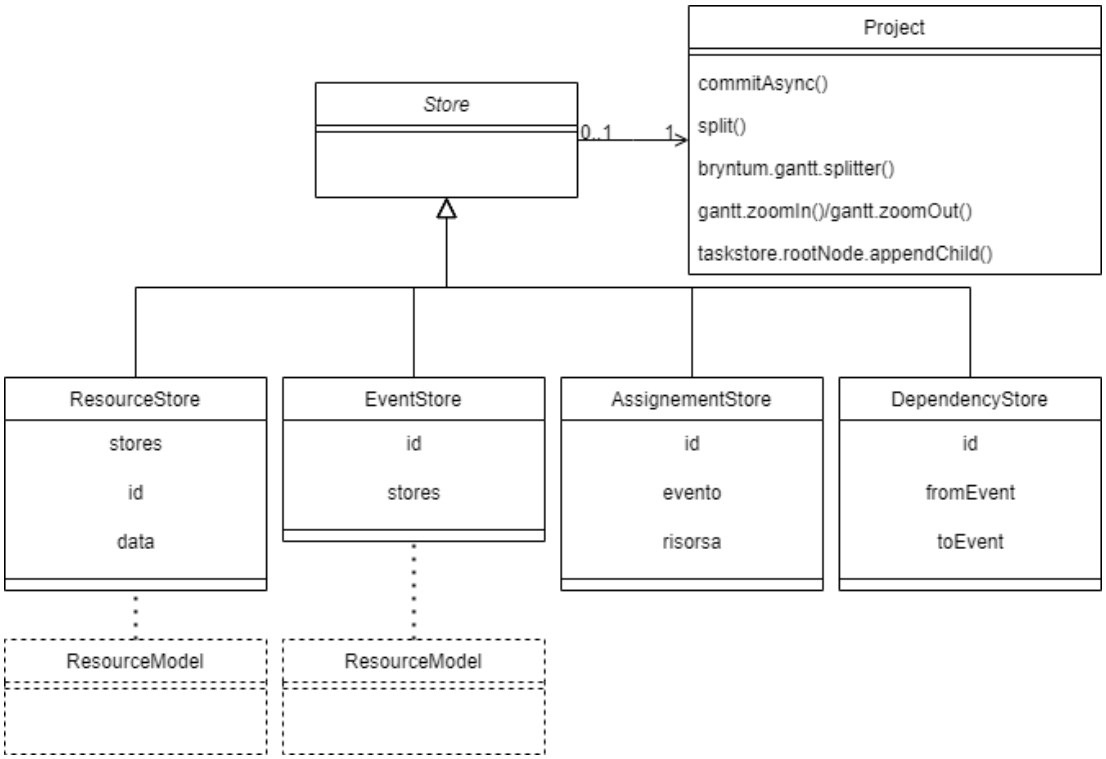

Figura 4.6: Classi d'interesse per lo sviluppo

Le parti fondamentali di lavoro nella codifica saranno mirate a:

- Rendere gli **store** disponibili allo Scheduler
- Calcolare le date e la durata tramite chiamate a funzioni, e poi chiamare il **await project.commitAsync()** in maniera asincrona
- Tenere i dati tra gli **store** aggiornati

Durante il normale utilizzo di base dell'interfaccia utente, non sarà necessario interagire molto con il **Project**. Dato che i dati devono essere locali secondo richiesta del tutor di azienda.
Il componente **Store** nel **Project** è necessario per renderizzare il diagramma Gantt/Scheduler e poi in seguito il l'istogramma.

4.4.2 Codifica

Primo mockup^G e iterazione

Subito dopo la primissima iterazione (come in figura 4.7) del prodotto si è notato che il diagramma "scheduler" era il metodo corretto per proseguire con la codifica dato che l'utilizzo e implementazione del diagramma "scheduler" offerto dalla Bryntum era più completo in termini di features rispetto al diagramma "Gantt".

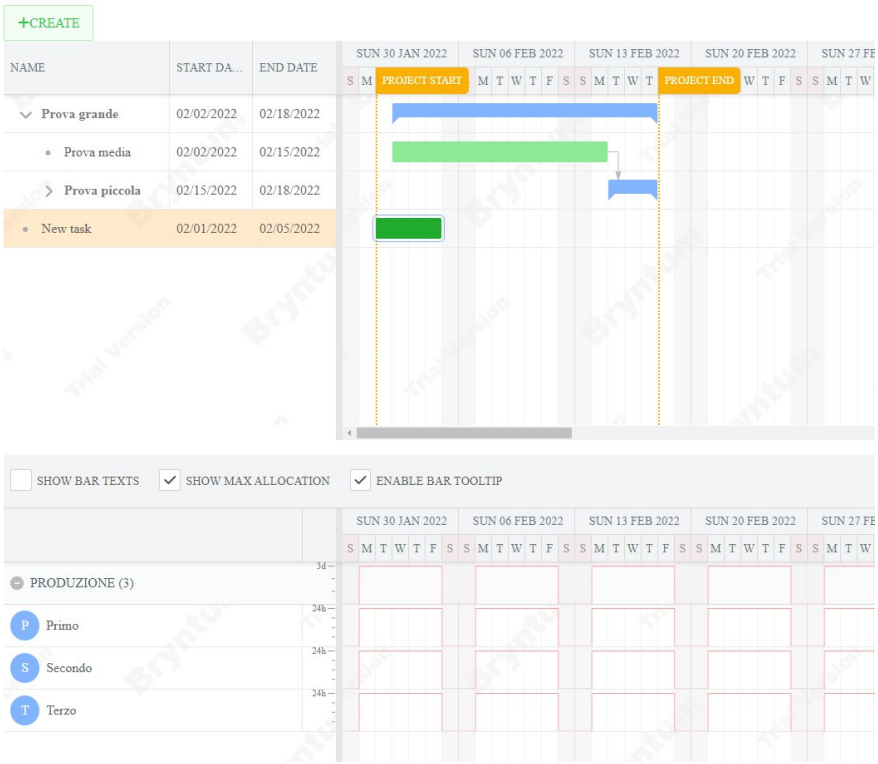

Figura 4.7: Primo prodotto Bryntum con gantt

Ciò però necessitava un sacrificio da parte della funzionalità: cioè di riuscire a offrire una percentuale per ogni attività e indicarne la completezza come nell'immagine 4.8.
Dopo un meeting con i tutori aziendali si è discusso e si è realizzato che la feature poteva essere messa come facoltativa e quindi si poteva proseguire con lo sviluppo del prodotto.

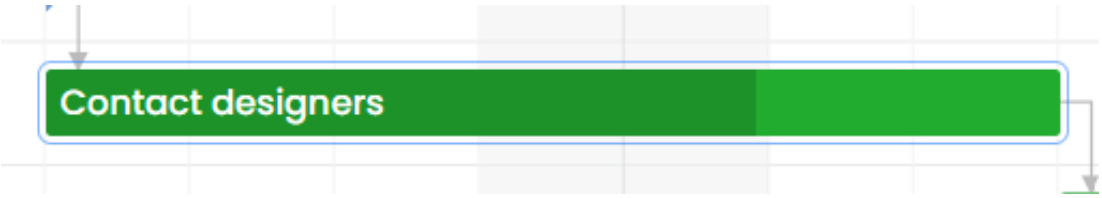

Figura 4.8: Esempio feature abbandonata passando da diagramma gant a scheduler

Implementazione due o più diagrammi simultanei

La prima parte dello sviluppo del prodotto è stata implementare la libreria in maniera tale che mostrasse uno scheduler Gantt e un Istogramma.
Come da immagine i due diagrammi sono uno sovrastante l'altro condividendo l'asse temporale (come in immagine 4.9). Ciò implica che anche una modifica degli assi crei una rielaborazione di entrambi i grafici.

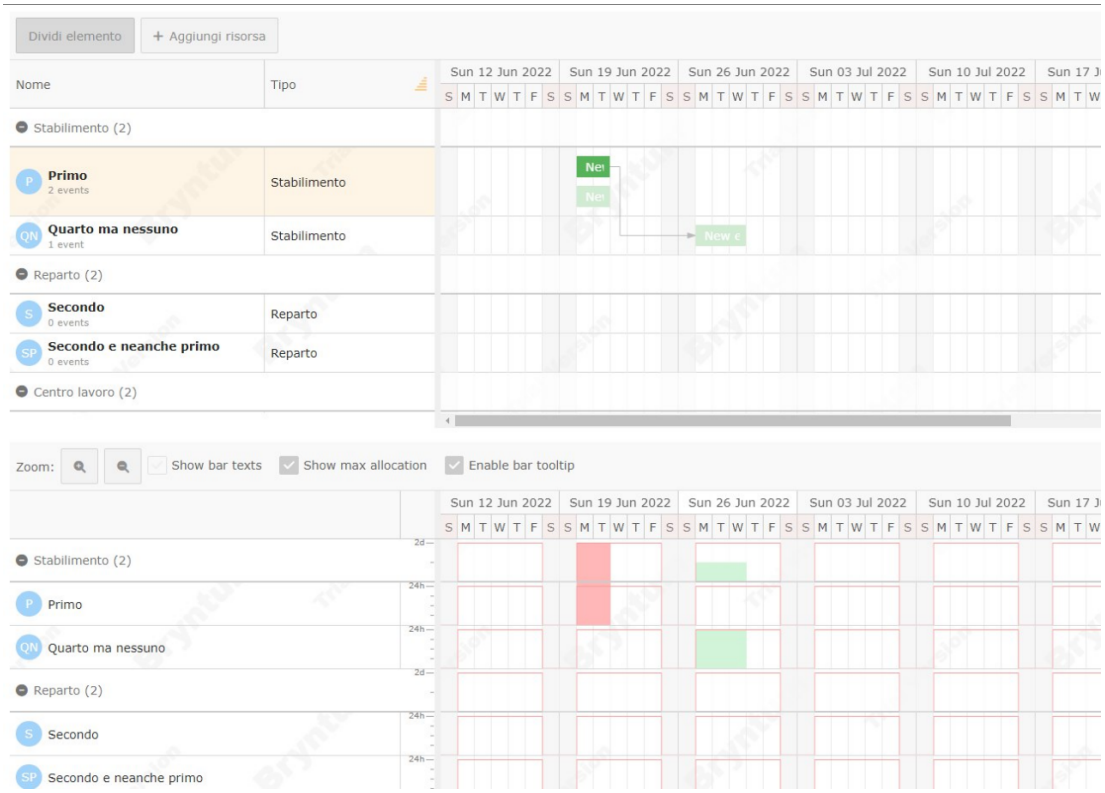

Figura 4.9: Interfaccia Bryntum con una attività divisa in due

Nella seconda parte ho codificato i primi requisiti funzionali:

- la creazione della funzione Split()
- Creazione di un sistema di dipendenze
- Eliminazione attività

Implementazione funzione Split()

La funzione **Split()** è importante per i committenti e, in quanto tale, una gran parte di studio è stata dedicata alla sua realizzazione.
Necessita infatti di una conoscenza profonda delle classi, dato che split deve suddividere una task in due attività diverse e possibilmente che rappresentassero la metà della task originale in termini temporali.

La parte che ha richiesto la maggior parte del tempo lavoro non è stata la creazione di una nuova attività, ma che le nuove attività rispettassero le previe dipendenze.

Figura 4.10: Dipendenze mantenute

Una volta modificato l'**EventStore()** è necessaria una chiamata asincrona al renderer di Gantt e dell' Istogramma, che viene normalmente attivato in maniera automatica da un observer dello **Store**. Per le dipendenze (se esistenti) ne viene estrapolato l'ID, salvato su una variabile temporanea per poi viene applicato nuovamente alle due attività create, esempio in figura 4.10.
Questo ID è stato estrapolato utilizzando una funzione apposita chiamata **GetStoreID()** utilizzata assieme a **GetSelectedID()**.
Ogni fine settimana veniva effettuata una presentazione da parte del sottoscritto ai tutori che valutano il prodotto in termini qualitativi e in base all'adempimento dei requisiti funzionali.
La eliminazione delle attività è stata compiuta tramite gli strumenti già presenti all'interno della libreria e quindi è stata la più semplice da implementare.
Un possibile rischio è che la funzione **DeleteEventStoreID()** non riesca ad eliminare le attività create le attività create da **Split()** cosa non riscontrata, dato che le attività generate da **Split()** e quelle generate dalla interfaccia grafica si comportano alla stessa maniera.

4.4.3 Pagina principale

La pagina principale del prodotto deve rendere chiaro le principali funzionalità obbligatorie del prodotto, che sono:

- Poter creare a proprio piacimento delle gerarchie come nella realtà di produzione (La gerarchia è Stabilimento, reparto e centro di lavoro).
- Divisione delle attività tra gerarchie (In cui uno stabilimento contiene reparti che a loro volta contengono dei centri di lavoro), in maniera tale che le attività di livelli inferiori appartengono a una supertask del livello superiore
- La capacità da parte dell'utente di suddividere un'attività in sottoattività sempre rispettando la gerarchia

La implementazione delle gerarchie sfortunatamente non ha portato i risultati voluti, infatti la implementazione di raggruppamenti di attività poteva avvenire solo una volta.
Infatti non si può nella libreria creare gruppi di gruppi di attività, quindi il vincolo di creare gerarchie non è stato rispettato appieno.

In ordine di suddividere in maniera appropriata la pagina tra istogramma e Scheduler viene utilizzata una funzione offerta della libreria chiamata **Bryntum.gantt.Splitter()** (da non confondersi con la funzione split() richiesta dal committente), un esempio di divisione tra i diagrammi è mostrato in 2.2 e 4.13.

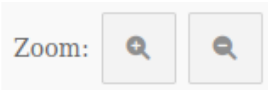

Figura 4.11: Pulsanti zoom

Per soddisfare i requisiti obbligatori di zoom e creazione task vengono implementate le funzioni asincrone **gantt.zoomOut()/gantt.zoomIn()** e **taskStore.rootNode.appendChild()** rispettivamente, visti in figura 4.11.

Uno studio della libreria e della implementazione di essa ha portato alla luce che il diagramma di Gantt era simile a uno scheduler, quindi i previ termini e funzioni possono essere utilizzati intercambiabilmente tra i diagrammi.

Figura 4.14: Istogramma con massimo numero di ore

Nelle immagini è mostrato il progresso della produzione Bryntum. Nel primo caso è presente uno scheduler con a fianco un istogramma dove vengono indicate le ore (figura 2.2).
Nella immagine finale in figura 4.13 è esposto l'ultimo prodotto sviluppato Bryntum. Come possiamo vedere esistono delle attività con relazioni e delle risorse collassabili. Nelle immagini 2.2 e 4.13 si notano sia lo scheduler nella parte superiore che un istogramma nella parte inferiore. Si nota inoltre come le risorse sono collassabili e come nella parte

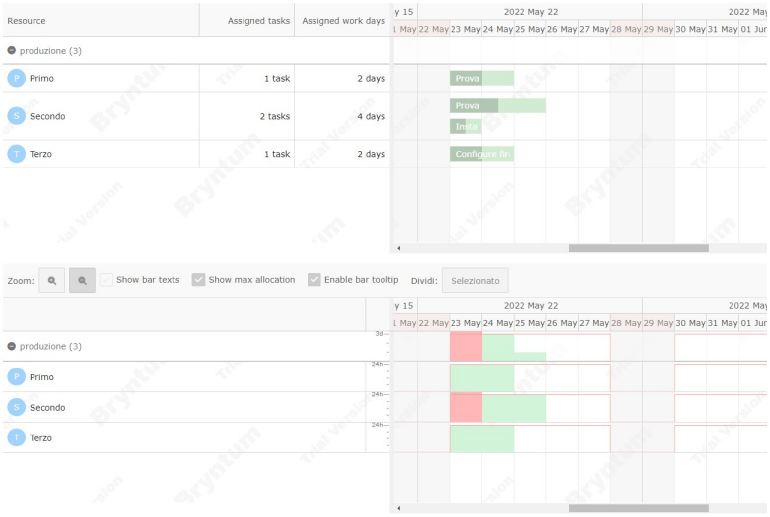

Figura 4.12: Implementazione dello scheduler con l'istogramma

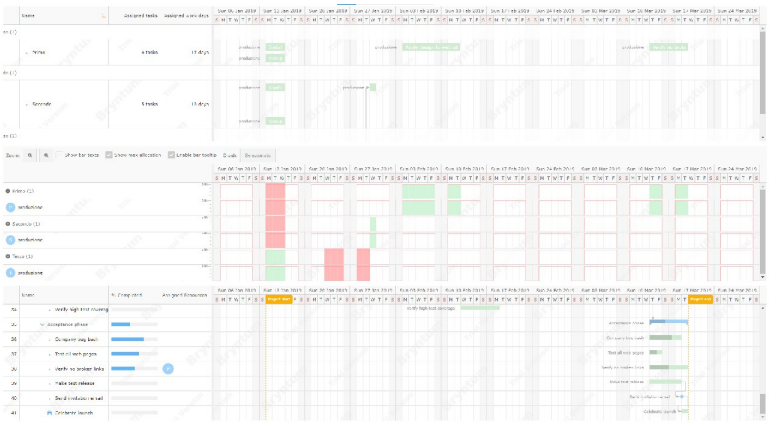

Figura 4.13: Implementazione dello scheduler, istogramma e gantt in contemporanea

della pagina dedicata all'istogramma esista una sezione in cui le ore assegnate sono più delle ore disponibili (esempio immagine 4.14), e che il tasto della divisione ha creato una attività divisa 4.15.

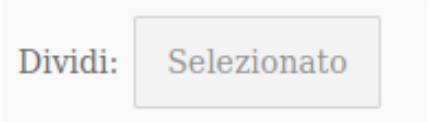

Figura 4.15: Pulsante utilizzato per dividere una attività dopo la selezione

4.4.4 Chiamate alla libreria

Nel seguente sequence diagram^G sono esposte le chiamate iniziali necessarie per la costruzione di entrambi gli istogrammi.
Come si può notare ci sono due chiamate alla libreria, una JavaScript e una CSS. Ogni qualvolta avviene una modifica si eseguono simili chiamate alla libreria per rielaborare i grafici.
Come si può notare dalla figura 4.16 la prima chiamata è la chiamata per la struttura HTML, la seconda è per il CSS e l'ultimo per il JavaScript. Le chiamate alla libreria sono necessarie per la costruzione della pagina.

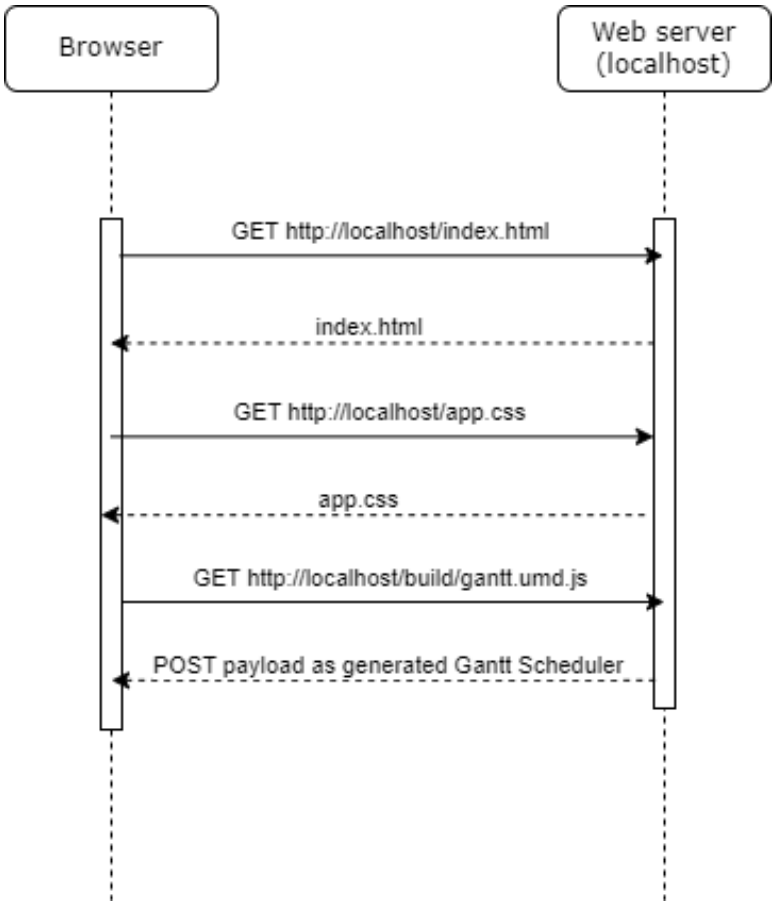

Figura 4.16: Sequence diagram delle prime chiamate necessarie

4.4.5 Problematiche rilevate

Le problematiche rilevate sono state svariate. La prima riguarda la implementazione della libreria, il sito indicato nelle fonti presentava pochi esempi inerenti a uno scheduler legato a istogramma, infatti la maggior parte degli esempi si concentrano su dei diagrammi di Gantt che non rispettavano a pieno i bisogni del committente.
La creazione di **split()** necessitava un'approfondita conoscenza della struttura delle classi che la libreria utilizza, la quale è stata un una problematica che ha causato dei rallentamenti nel tempo di analisi, e si sono rispecchiate negativamente sul tempo dedicato alla codifica.
Una problematica secondaria è che lo scheduler e l'istogramma fanno parte di due prodotti che generalmente non interagiscono tra di loro e utilizzano due tipi chiamate ai dati diverse.
Unire i due tipi di prodotti diversi della Bryntum ha presentato dei rallentamenti soprattutto dovuti alla stessa mancanza di esempi nel sito, con una descrizione delle librerie e delle funzioni carente.

4.5 DevExpress

Nel caso della libreria DevExpress sono state presentate immediate difficoltà durante l'esperienza di stage. La produzione è stata divisa in due parti:

4.5.1 Progettazione

La prima parte della esperienza di stage con la libreria DevExpress è stata uno studio della libreria dove è apparso immediatamente che alcuni requisiti funzionali non potevano essere rispettati.
Lo sviluppo del prodotto è proseguito comunque dato che la azienda aveva già dei prodotti con DevExpress e aveva già acquistato delle funzionalità dalla software house.
I vincoli non rispettabili sono i seguenti:

- Relazione tra attività
- Istogramma
- Modifica responsive della dimensione temporale
- Gerarchia tra centri di produzione
- Raggruppamento di diversi centri di produzione

Dopo una attenta analisi da parte del sottoscritto, si nota che la libreria DevExpress presenta delle importanti limitazioni nella creazione dello scheduler, in quanto la libreria come "scheduler" non si riferisce a un organigramma con delle dipendenze, ma una semplice assegnazione di task a delle risorse tramite una visualizzazione calendarizzata.
Per arginare la mancanza di un effettivo metodo reactive per la modifica dell'asse temporale ho iterato sulla già esistente visualizzazione a viste, modificando la vista "a calendario" che normalmente si visualizza nello scheduler mensile a una vista "a linea del tempo" più consono al diagramma desiderato.
È stato creato un primo disegno della pagina assecondando le limitazioni imposte dalla libreria in figura 4.17.

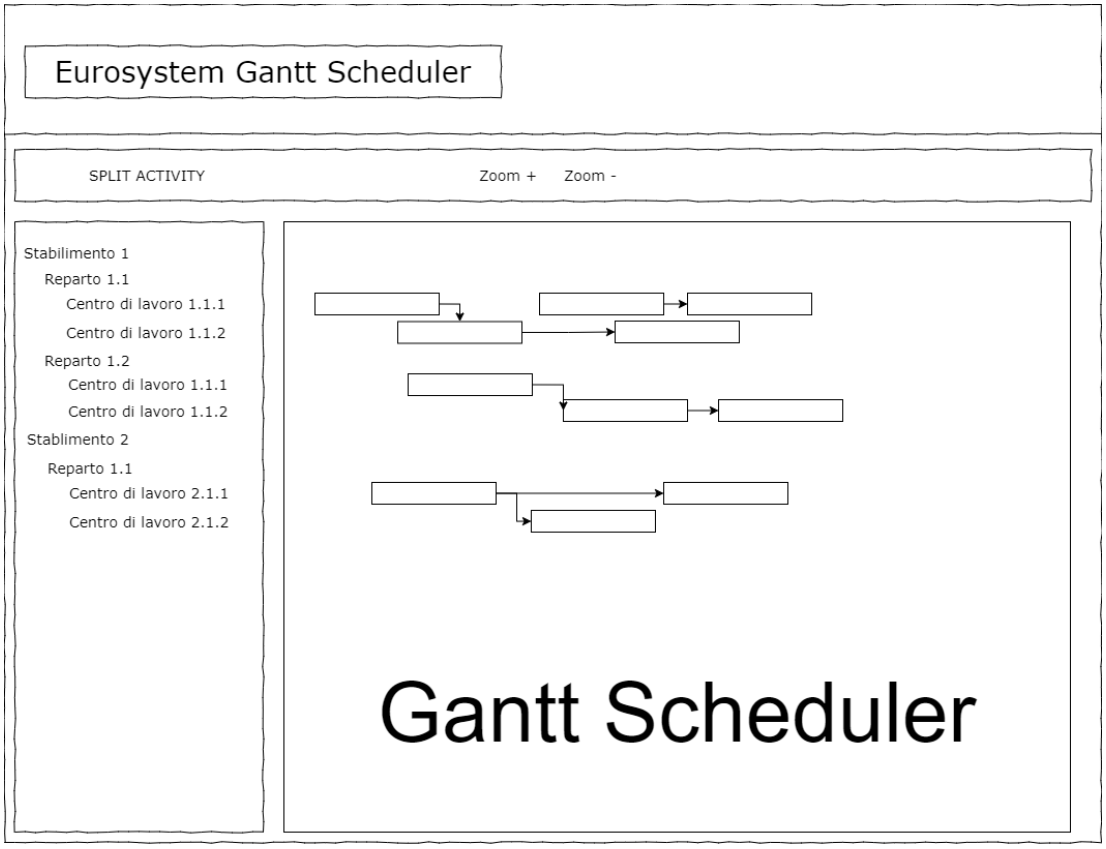

Figura 4.17: Mockup Devexpress

4.5.2 Codifica

La libreria DevExpress viene strutturata in maniera non dissimile da un progetto complesso, modulare ed espandibile, al contrario delle librerie precedenti, che si basavano su implementazioni monolitiche e direttamente sul corpo della pagina dove si visualizzava il grafico.

Essendo limitato alla visualizzazione delle viste predefinite da parte della libreria senza molto spazio per modifica, la implementazione più vicina ai requisiti richiesti era una visualizzazione di una timeline dove ogni task veniva assegnata alla risorsa.

Figura 4.18: Pulsanti per cambio delle viste

Dato che un requisito è creare delle viste con diverse scale temporali, ho creato diverse viste con scale diverse nell'asse temporale (scambiabili tramite pulsante a immagine 4.18).
Quest'ultima è soluzione accettabile ma non adeguata come la soluzione proposta da Bryntum, che invece offriva una visualizzazione attiva della linea temporale.
Per realizzare le viste sono state utilizzate svariate chiamate a funzione in combinazione mostrate in figura 4.19:

- **timeCellTemplate()**: All'interno di DevExpress e soprattutto nello schedule vengono create delle celle. Le celle sono delle istanze grafiche dell'attività che vengono assegnate ad ogni singola risorsa. Quando le celle vengono create esistono su una dimensione temporale modificabile mediante timeCellTemplate().

- **dateCellTemplate()**: Una volta definita la dimensione base minima delle celle è stato semplicemente necessario creare una scala personalizzata per la visualizzazione delle informazioni su diagramma. Per fare questo è stata utilizzata questa dateCellTemplate().

- **timeCellRender()**: In certe casistiche è stato necessario utilizzare questa funzione per renderizzare celle da una visita all'altra. È necessario utilizzare la funzione timeCellRender() quando le celle non sono state completamente create. Un esempio potrebbe essere creare un'attività nella vista settimana ma non confermarla e poi passare nella vista mese aggiustarla e poi confermarla, o viceversa.

- **currentView()**: La funzione è necessaria a scopo di trovare la vista su cui il prodotto si trova per poter agire in maniera più mirata durante la creazione delle celle di attività. Ad esempio può essere che nella visualizzazione mensile non è strettamente necessario inserire gli orari quando si inserisce l'attività tramite trascinamento. Mentre in una visualizzazione settimanale è convenzione (consiglio da parte del committente) che sia necessario inserire delle date precise.

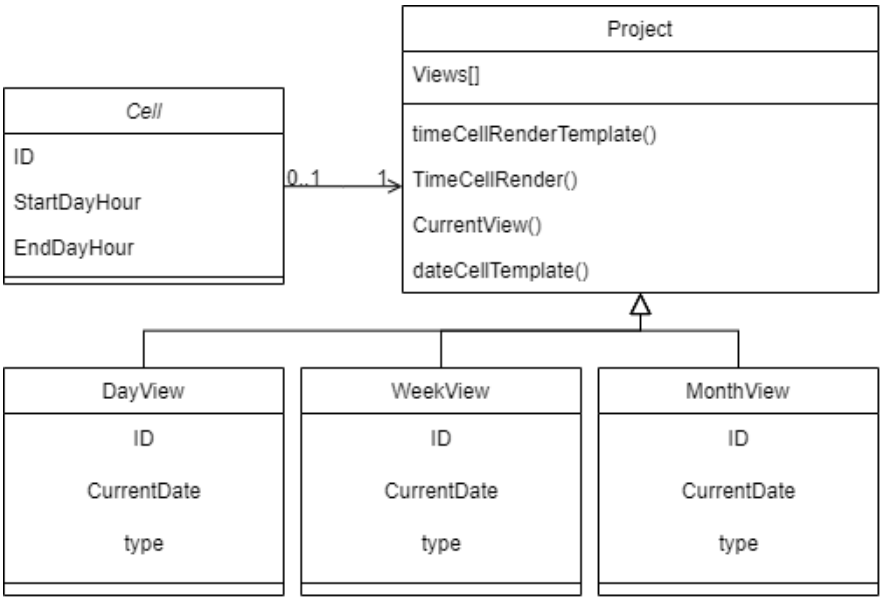

Figura 4.19: Struttura classi per le viste DevExpress

Le seguenti immagini rappresentano il lavoro completato di analisi e realizzazione di una libreria DevExpress secondo i requisiti imposti dal committente. La immagine 4.20 mostra una vista settimanale, dove un utente può inserire le celle relative alle attività nella dimensione settimanale.
La seconda immagine 4.21 mostra invece la vista cambiata da vista "calendario" (come in figura 4.22) come dal sito esempio DevExpress a una vista "linea temporale".

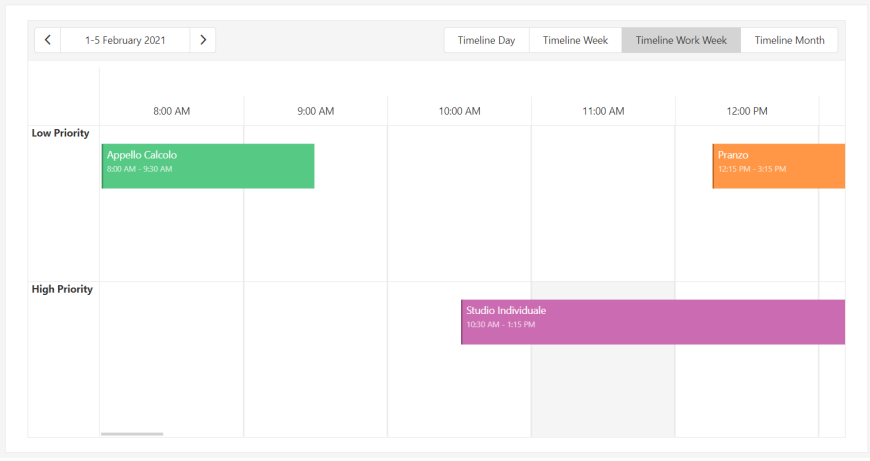

Figura 4.20: Scheduler DevExpress Settimana

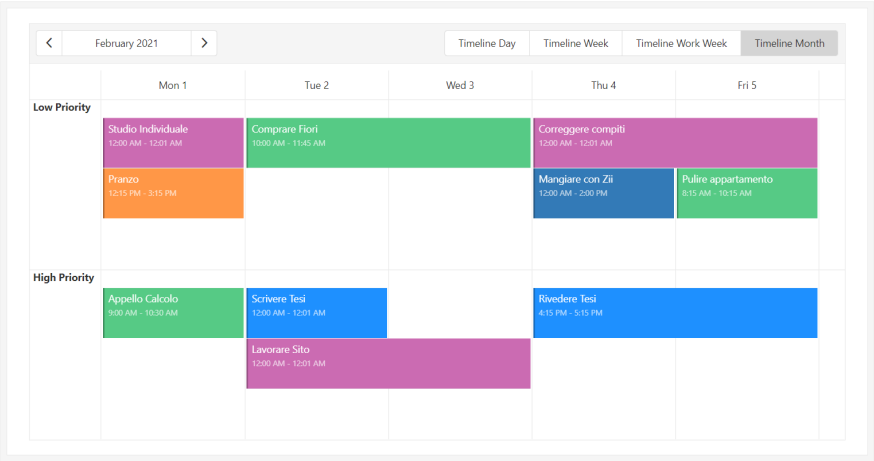

Figura 4.21: Scheduler DevExpress Mese

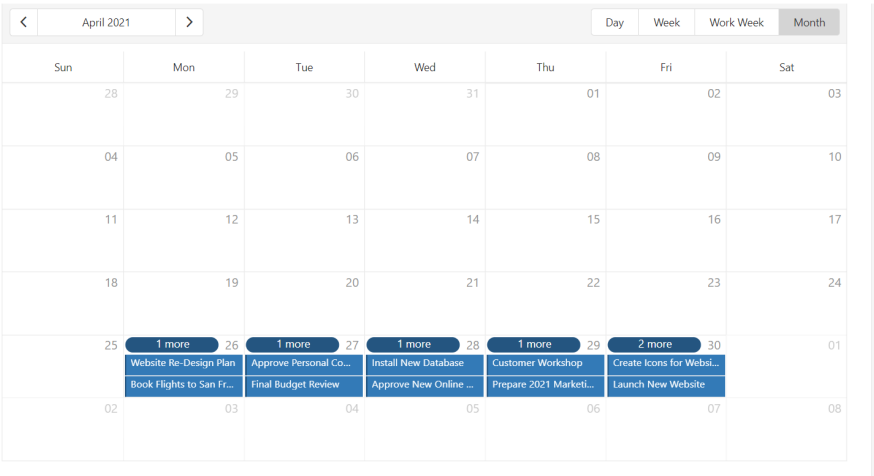

Figura 4.22: Scheduler DevExpress Mese prima delle modifiche

4.6 NetTronic

Questa piccola parentesi è per la considerazione di utilizzare la libreria NetTronic. Quest'ultima libreria è stata valutata nelle prime due giornate. Esistono due varianti del programma, una che utilizza Windows forms (quindi un file .exe) e una web (figura 4.23).
NetTronic inizialmente si era dimostrata molto promettente, in quanto forniva una suite di funzioni che avrebbero soddisfatto le specifiche del progetto.
Sul sito è presente sia la versione eseguibile (.exe) che la libreria per lo sviluppo web (figura 4.23). In quanto interessati alla libreria abbiamo notato che è possibile accedere alla libreria solo con un meeting da remoto con un corrispondente NetTronic. Dato che la chiamata poteva essere effettuata solo un mese più avanti, la NetTronic è stata scartata.

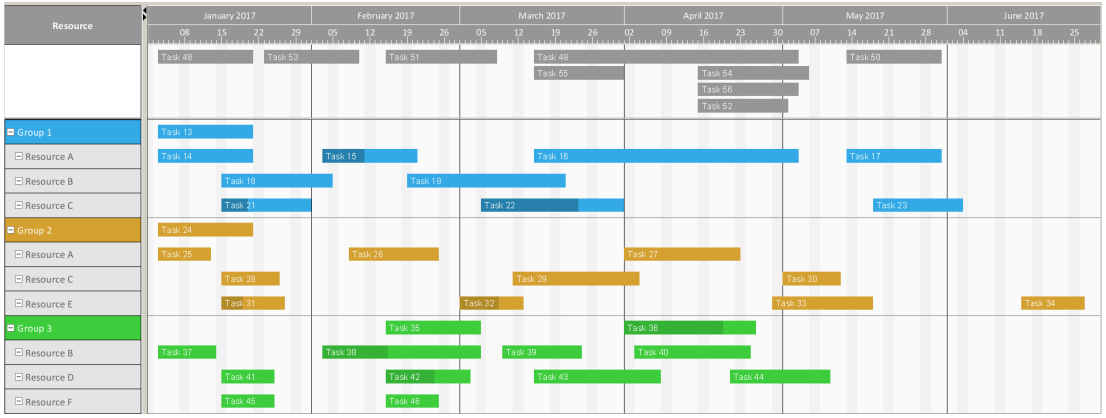

Figura 4.23: Un esempio di un possibile prodotto NetTronic

4.7 Design pattern

Facade

Facade è una classe che fornisce un'interfaccia semplice a un sottosistema complesso che contiene molte parti mobili. In questo caso la libreria possiede un sistema parecchio completo ma complesso. Per questo sono implementate funzioni in grado di usarne le funzionalità senza incorrere in problemi di complessità.

Observer

Essendo il sistema aggiornato in maniera asincrona, la funzione **Split()** ogni divisione di due **Events** chiama **eventStore.project.commitAsync()** in maniera asincrona, per poi aggiornare l'intero **Project** con **project.commitAsync()**.

4.8 Project menagement

Scrum

Scrum è un framework mirato al project menagement per lo sviluppo del software. Il prodotto viene suddiviso in degli obiettivi che possono essere completati entro iterazioni temporali, chiamate sprint, di durata equivalente a una o massimo due settimane.
Lo Scrum prevede la valutazione dei progressi nelle riunioni giornaliere di 15 minuti o meno, chiamate "daily scrum" (questo in genere avveniva quando il tutore aziendale era presente la mattina).
Alla fine dello sprint, il team tiene altri due incontri: uno "sprint review" destinato a dimostrare il lavoro svolto e ottenere feedback, e uno sprint retrospettivo destinato a consentire al team di riflettere e migliorare.
Durante la riunione ho entrambe le riunioni, presento il prodotto settimanale e ricevo feedback sui campi dove potrei migliorare.

Capitolo 5

Conclusioni

5.1 Generale

L'esperienza di stage è avvenuta in diverse parti: prima parte è stata l'introduzione dell'azienda, dove mi sono confrontato con le tecnologie usate e i tutori aziendali.
Mi è stato fornito il materiale necessario per migliorare le mie abilità di software engineering, degli obbiettivi da raggiungere e consigli sul come meglio approcciare le possibili difficoltà future.
Nella seconda parte ho lavorato su una serie di prodotti creati assieme ai tutori a scopo di capire quali librerie fossero più adatte ai bisogni dell'azienda.
La parte dello sviluppo principale è avvenuta stata l'implementazione della librerie Bryntum, DevExpress e NetTronic rispettivamente.
Durante lo sviluppo della prima (la libreria più completa e quella che alla fine è stata selezionata come la libreria finale) c'è stato un lungo percorso di confronto tra il tutore e le mie proposte per risolvere le svariate problematiche.
La libreria DevExpress mi ha permesso di sperimentare com'è utilizzare una libreria incompleta e come infine colmarne il divario tecnico.
Nell'ultima parte è avvenuta una considerazione generale dello stato dei prodotti.

5.2 Scelta libreria

La scelta della libreria è avvenuta tramite un meeting finale tra me e i tutori. È chiaro che una libreria è emersa come più vicina ai bisogni dell'azienda EuroSystem e di conseguenza si è deciso che la Bryntum era la più completa.
Nello specifico sono elencati sotto i requisiti funzionali e qualitativi rispettati.
È necessario sottolineare che la NetTronic avrebbe potuto avvicinarsi, ma l'impossibilità di accedere alle funzioni e di provare gratuitamente la libreria ha sicuramente fermato sul nascere il possibile sviluppo.
Infine la DevExpress ha dimostrato che è distante da una visualizzazione "scheduler" vicina a quello che EuroSystem ha richiesto, necessita infatti di lavoro

aggiuntivo per colmarne le mancanze.

5.3 Rispetto requisiti funzionali

5.3.1 Bryntum

Requisiti Funzionali

- **RF[OBB]1**: Il prodotto deve essere utilizzabile da interfaccia web;
  Soddisfatto
- **RF[OBB]2**: Il prodotto deve poter creare ed eliminare delle attività per ogni centro lavorativo;
  Soddisfatto
- **RF[OBB]2**: Bisogna essere in grado di creare o rimuovere utenti;
  Soddisfatto
- **RF[OBB]2**: Deve essere presente un istogramma per mostrare a colpo d'occhio le ore lavorative per ciascun individuo;
  Soddisfatto
- **RF[OBB]3**: Le attività devono essere collegabili tra di loro;
  Soddisfatto
- **RQ[OBB]3.1**: Le attività devono essere divisibili;
  Soddisfatto: Tramite il tasto dividi
- **RF[OBB]3.3**: Tale divisione deve portare a una divisione delle previe dipendenze, lasciando l'attività virtualmente uguale in termini di flusso lavorativo;
  Soddisfatto
- **RF[OBB]3.4**: Eliminare una attività deve eliminare anche le dipendenze o dare avviso della loro imminente eliminazione;
  Soddisfatto
- **RF[OBB]4**: Deve essere tutto contenuto in una istanza web, senza utilizzo di file temporanei (come ad esempio cookies).
  Soddisfatto
- **RF[OBB]5**: Deve esistere una gerarchia stabilimento/ reparto/centro di lavoro;
  Parzialmente soddisfatto: Esistono solo un raggruppamento all'interno delle attività, non è possibile creare gruppi di gruppi
- **RF[OBB]5.1**: Le gerarchie devono essere flessibili al punto di poter trasferire una attività da un centro di lavoro all'altro, o suddividere una attività per operazioni in parallelo di vari centri di lavoro;
  Soddisfatto

- **RF[OBB]6**: Deve essere possibile interagire con la risoluzione della linea temporale, come visualizzare in mesi, settimane o giorni;
  Soddisfatto: Tramite tasti + e -

- **RF[DES]7**: Le attività raggruppate possono essere collassate per una visione più agevole.
  Parzialmente soddisfatto: Non è possibile raggruppare gruppi di risorse

- **RF[DES]8**: L'istogramma delle ore di lavoro mostra quando le ore assegnate sono superiori alle ore assegnabili;
  Soddisfatto

- **RF[DES]9**: La funzione di divisione delle attività deve suddividere anche a livello temporale, cosicché una attività si suddivida in due mezze attività;
  Soddisfatto

Obiettivi qualitativi

- **QF[OBB]1**: Il codice deve essere condiviso sulla cartella pubblica;
  Soddisfatto

- **QF[OBB]2**: La pagina web deve essere responsive^G;
  Soddisfatto

- **QF[OBB]3**: La pagina web deve essere intuitiva;
  Soddisfatto

- **QF[OBB]4**: La pagina web deve essere mobile-friendly^G;
  Non soddisfatto: Le librerie sono concentrate sull'utilizzo da parte di aziende, dove le interazioni utente sono effettuate da PC fisso.

5.3.2 DevExpress

La seconda libreria presa seriamente in considerazione da parte dei relatori di azienda è stata la DevExpress.
Come notato nel capitolo precedente. Presenta importanti carenze su requisiti molto importanti.

Requisiti Funzionali

- **RF[OBB]1**: Il prodotto deve essere utilizzabile da interfaccia web;
  Soddisfatto
- **RF[OBB]2**: Il prodotto deve poter creare ed eliminare delle attività per ogni centro lavorativo;
  Parzialmente soddisfatto
  Non è possibile raggruppare le attività
- **RF[OBB]2**: Bisogna essere in grado di creare o rimuovere utenti;
  Soddisfatto
- **RF[OBB]2**: Deve essere presente un istogramma per mostrare a colpo d'occhio le ore lavorative per ciascun individuo;
  Non Soddisfatto
  DevExpress non presenta un istogramma per accompagnare il diagramma principale
- **RF[OBB]3**: Le attività devono essere collegabili tra di loro;
  Non Soddisfatto
  Non è possibile aggiungere alcun tipo di dipendenza
- **RQ[OBB]3.1**: Le attività devono essere divisibili;
  Non Soddisfatto
  Non esistono funzioni in grado di dividere per grandezze definite dallo sviluppatore. È possibile che si possa implementare, ma la implementazione di funzioni ad hoc va oltre lo scopo dello stage.
- **RF[OBB]3.3**: Tale divisione deve portare a una divisione delle previe dipendenze, lasciando l'attività virtualmente uguale in termini di flusso lavorativo;
  Non Soddisfatto
- **RF[OBB]3.4**: Eliminare una attività deve eliminare anche le dipendenze o dare avviso della loro imminente eliminazione;
  Non Soddisfatto
- **RF[OBB]4**: Deve essere tutto contenuto in una istanza web, senza utilizzo di file temporanei (come ad esempio cookies).
  Soddisfatto

- **RF[OBB]5**: Deve esistere una gerarchia stabilimento/ reparto/centro di lavoro;
  Non Soddisfatto: Non è possibile stanziare delle dipendenze

- **RF[OBB]5.1**: Le gerarchie devono essere flessibili al punto di poter trasferire una attività da un centro di lavoro all'altro, o suddividere una attività per operazioni in parallelo di vari centri di lavoro;
  Parzialmente soddisfatto: È possibile passare attività da una risorsa a un altra, ma non è possibile stanziare delle gerarchie.

- **RF[OBB]6**: Deve essere possibile interagire con la risoluzione della linea temporale, come visualizzare in mesi, settimane o giorni;
  Parzialmente soddisfatto: Sono state create delle viste che permettono una visualizzazione giornaliera, settimanale e mensile

- **RF[DES]7**: Le attività raggruppate possono essere collassate per una visione più agevole.
  Non Soddisfatto

- **RF[DES]8**: L'istogramma delle ore di lavoro mostra quando le ore assegnate sono superiori alle ore assegnabili;
  Non Soddisfatto: Non è possibile avere un istogramma che coincide con le informazioni dello scheduler in contemporanea

- **RF[DES]9**: La funzione di divisione delle attività deve suddividere anche a livello temporale, cosicché una attività si suddivida in due mezze attività;
  Non Soddisfatto: Nella libreria non esiste tale funzione.

Obiettivi qualitativi

- **QF[OBB]1**: Il codice deve essere condiviso sulla cartella pubblica;
  Soddisfatto

- **QF[OBB]2**: La pagina web deve essere responsive^G;
  Non soddisfatto: A causa dell'approccio a viste il prodotto non corrisponde ai canoni di un web design responsive.

- **QF[OBB]3**: La pagina web deve essere intuitiva;
  Soddisfatto

- **QF[OBB]4**: La pagina web deve essere mobile-friendly^G;
  Non soddisfatto: Le librerie sono concentrate sull'utilizzo da parte di aziende, dove le interazioni utente sono effettuate da PC fisso.

5.4 Valutazioni personali

L'esperienza in azienda è stata per la maggior parte positiva per due principali fattori.
Il primo riguardante la libertà con la quale ho potuto affrontare i problemi dati e il secondo è inerente alll'esposizione di quello che ho prodotto ogni fine settimana; questo processo mi ha spronato a migliorare le mia abilità di speaker e le mie soft skills.
Durante il percorso in azienda e la scrittura di questo elaborato ho omesso svariate parti del mio percorso specialmente nelle prime settimane poiché non strettamente inerenti allo sviluppo di un singolo prodotto.
Infatti nelle prime settimane è avvenuto un attento studio della libreria jQuery. Ho infatti realizzato diversi piccoli prodotti secondo le specifiche limitate date dai relatori d'azienda.
Lavorare in azienda mi ha aiutato a migliorare le mie capacità relative allo sviluppo web che rispetti i vari vincoli dati dai tutori. Questi ultimi inoltre mi hanno aiutato a comprendere il lavoro in azienda e come in genere viene strutturato un software all'interno di una realtà professionale.
Inoltre il lavoro all'interno di una realtà aziendale di dimensioni medio-grandi come EuroSystem ha messo in prospettiva il tipo di possibilità di lavoro che si potrebbero presentare in futuro.
Nella realtà aziendale esiste un complesso tipo di organizzazione che viene creato attorno ai vari componenti dell'azienda. Un esempio potrebbe essere la registrazione e integrazione dei vari prodotti (Es. Teams) con l'account aziendale e come esso possa essere utilizzato per organizzare meeting e ticket. Il lavoro in azienda mi ha offerto una visione olistica dei funzionamenti aziendali e degli gli strumenti per poter in futuro relazionarmi ad aziende simili ad essa.

5.5 Termine stage e possibilità future

Al termine dello stage è stato esposto l'ultimo diagramma di Gantt DevExpress e si è fatta una valutazione totale di tutte le parti dell'attività e della produzione finale.
Si è ragionato su una possibile futura implementazione delle librerie. Esse infatti possono essere espanse attraverso altre ore lavoro probabilmente assegnabili a dei dipendenti della azienda, e quindi, all'implementazione persino di funzioni future all'interno della DevExpress. La possibilità di sviluppo delle funzioni all'interno della libreria DevExpress è possibile dato che l'azienda EuroSystem già possiede i diritti a parti della libreria, le quali possono essere espanse nel futuro.
So è pensato di lavorare con l'azienda DevExpress informandoli dei bisogni dell'EuroSystem nel loro forum online che tengono attivo e aggiornato.

5.6 Conoscenze Acquisite

Durante il percorso di stage sono state guadagnate delle competenze inerenti allo sviluppo di applicazioni web del framework ASP.NET, jQuery, HTML e CSS. Si è guadagnata una maggiore comprensione del modello MVC e, in maniera più importante, sono state sviluppate molte soft skill. Quest'ultime sono state migliorate durante il percorso di stage e le interazioni interpersonali che sono avvenute tra me stesso e i tutori. Queste abilità sono state sviluppate ulteriormente durante i vari meeting non solo con i tutori ma anche con l'intero staff EuroSystem. Nel complesso è stata una esperienza positiva e mi ha aiutato a maturare in un sviluppatore software più completo.

Capitolo 6

Glossario

- **B**
  - **Bryntum**: Bryntum è un azienda fondata nel 2009 e da allora fornisce componenti di pianificazione JavaScript di alta qualità per il web.
- **C**
  - **Casi d'uso**: Il caso d'uso in informatica è una tecnica usata nei processi d'ingegneria del software per effettuare in maniera esaustiva e non ambigua, la raccolta dei requisiti al fine di produrre software di qualità. Essa consiste nel valutare ogni requisito focalizzandosi sugli attori che interagiscono col sistema, valutandone le varie interazioni.
  - **Controllo versione**: Il controllo versione (version control), in informatica, è la gestione di versioni multiple di un insieme di informazioni: gli strumenti software per il controllo versione sono ritenuti molto spesso necessari per la maggior parte dei progetti di sviluppo software o documentali gestiti da un team collaborativo di sviluppo o redazione.
- **D**
  - **Database**: In informatica, una base di dati o banca dati (in inglese database, comune anche in italiano) è una collezione di dati organizzati immagazzinata e accessibile per via elettronica.
  - **Design Pattern**: Design pattern (traducibile in lingua italiana come schema progettuale, schema di progettazione, o schema architetturale), in informatica e specialmente nell'ambito dell'ingegneria del software, è un concetto che può essere definito "una soluzione progettuale generale ad un problema ricorrente". Si tratta di una descrizione o modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software, ancor prima della definizione dell'algoritmo risolutivo della parte computazionale. È un approccio spesso efficace nel contenere o ridurre il debito tecnico.

  - **DevExpress**: DevExpress è una piattaforma di sviluppo software per sviluppatori .NET e Visual Studio con una raccolta di componenti utilizzati per la creazione e la distribuzione di app su piattaforme Windows, Web e mobili.
  - **DOM**: In informatica il Document Object Model (spesso abbreviato come DOM), è una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti, utilizzato per la rappresentazione di elementi grafici in una pagina web.

- **F**
  - **Feature**: Nel software, il termine feature indica una caratteristica distintiva di un elemento software (ad esempio, prestazioni, portabilità o funzionalità).
  - **Framework**: in informatica e specificamente nello sviluppo software, un framework è un'architettura logica di supporto (spesso un'implementazione logica di un particolare design pattern) sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.
  - **Front-end**: Il front-end in informatica si intendono in una pagina web le parti visibili all'utente e quelle con cui si può interagire.

- **G**
  - **Diagramma di Gantt**: Un diagramma di Gantt è un grafico a barre orizzontali utilizzato nella gestione dei progetti per rappresentare visivamente un piano di progetto nel tempo. I diagrammi di Gantt mostrano in genere la sequenza temporale e lo stato, nonché chi è responsabile, per ogni attività del progetto.

- **I**
  - **IDE**: Un ambiente di sviluppo integrato (in inglese integrated development environment ovvero IDE, anche integrated design environment o integrated debugging environment, rispettivamente "ambiente integrato di progettazione" e "ambiente integrato di debugging"[senza fonte]), in informatica, è un software che, in fase di programmazione, supporta i programmatori nello sviluppo e debugging del codice sorgente di un programma.
  - **Industria 4.0**: L'Industria 4.0 sta rivoluzionando il modo in cui le aziende producono, migliorano e distribuiscono i loro prodotti. I produttori stanno integrando nuove tecnologie, tra cui Internet of Things (IoT), cloud computing e analisi, intelligenza artificiale e apprendimento automatico nei loro impianti di produzione e in tutte le loro operazioni.

- **L**
  - **Libreria**: In informatica, una libreria, è un insieme di funzioni o strutture dati predefinite e predisposte per essere collegate ad un programma software attraverso un opportuno collegamento.
  - **Linguaggio Dinamico**:Non esiste una definizione universalmente accettata per i linguaggi dinamici. Possiamo definire che qualsiasi linguaggio di programmazione che consente la modifica delle variabili nel Run-time, è un linguaggio dinamico.
  - **Linguaggio Interpretato**: Un linguaggio interpretato è un linguaggio che necessita di un interprete per essere eseguito. Un interprete, in informatica e nella programmazione, è un programma in grado di eseguire altri programmi a partire direttamente dal relativo codice sorgente scritto in un linguaggio di alto livello.
- **N**
  - **Node.js**: Node.js è un runtime system open source multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript.
- **M**
  - **Mockup**: Un mockup, o mock-up, è una realizzazione a scopo illustrativo o meramente espositivo di un oggetto o un sistema, senza le complete funzioni dell'originale.
- **O**
  - **Open Source**: Con open source in informatica, si indica un sistema di sviluppo software decentralizzato basato sulla condivisione dei file sorgenti (source).
  - **Organigramma**: L'organigramma è la rappresentazione grafica della struttura di una organizzazione.
- **P**
  - **Proof of Concept**: Il Proof of Concept (POC) è la prova ottenuta da un progetto pilota, che viene eseguito per dimostrare che un'idea di prodotto, un piano aziendale o unpiano di progettoè fattibile. Ad esempio, nello sviluppo di farmaci, gli studi clinici vengono utilizzati per raccogliere prove di concetto per un prodotto finale.
- **R**
  - **RaspberryPI**: Il Raspberry Pi è un computer a scheda singola. La scheda è stata progettata per ospitare sistemi operativi basati sul kernel Linux.

  - **Requisito funzionale**: Un requisito funzionale (F, RF, o FR dall'inglese "functional requirement"), nell'ambito dell'ingegneria del software, è un requisito che definisce una funzione di un sistema di uno o più dei suoi componenti, definendone la tipologia degli ingressi e delle uscite, nonché il comportamento.
  - **Responsive**: Il Responsive Web Design riguarda l'utilizzo di HTML e CSS per ridimensionare, nascondere, ridurre o ingrandire automaticamente, un sito Web, per renderlo adatto a tutti i dispositivi.

- **S**

  - **Sequence diagram**: Un Sequence Diagram (in italiano: Diagramma di sequenza) è un diagramma previsto dall'UML utilizzato per descrivere uno scenario.
  - **Scheduler**: Un prodotto software che consente a un'azienda di pianificare e tenere traccia delle attività in azienda.

- **U**

  - **UML**: In ingegneria del software, UML (Unified Modeling Language, "linguaggio di modellizzazione unificato") è un linguaggio di modellazione e di specifica basato sul paradigma orientato agli oggetti.

- **J**

  - **jQuery**: jQuery è una libreria JavaScript per applicazioni web. Nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi in una pagina web.
  - **JSON**: Il JSON è un formato di interscambio di dati, ed è spesso comparato con l'XML che però è un linguaggio di markup. Entrambi non hanno un sistema di rappresentazione dei dati binari, per cui è compito del programmatore adottare convenzioni appropriate (es. Base64) per convertire i dati binari in forma testuale.

Capitolo 7

Fonti

- **Wikipedia**:
  https://it.wikipedia.org/
- **Bryntum**:
  https://bryntum.com/
- **Esempio Bryntum**:
  https://bryntum.com/products/schedulerpro/examples/resourcehistogram/
- **Bryntum libreria**:
  https://bryntum.com/products/schedulerpro/docs/api
- **Eurosystem**:
  https://eurosystem.it/
- **DevExpress**:
  https://www.devexpress.com/
- **DevExpress Scheduler**:
  https://demos.devexpress.com/ASPNetCore/Demo/Scheduler/Timelines/NetCore/Light/
- **Documentazione ASP.NET**:
  https://dotnet.microsoft.com/en-us/learn/aspnet/architecture
- **stackoverflow**:
  https://stackoverflow.com/questions/29594105/mvc-is-it-model-to-view-or-controller-to-view
- **HTML e CSS**:
  https://www.w3.org/standards/webdesign/htmlcss
- **IBM**:
  https://www.ibm.com/topics/industry-4-0