Kanban Board

PROGETTO DI INGEGNERIA DEL SOFTWARE

INTRODUZIONE AL SISTEMA

Architettura generale

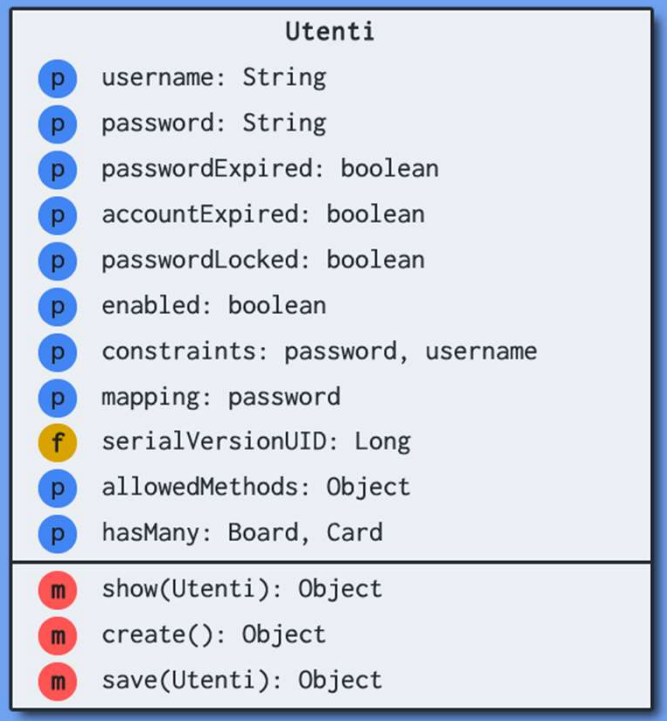
Di seguito verrà esposta l’architettura generale del sistema, con diagrammi delle classi e diagrammi di sequenza.

La classe utenti, rappresenta colui che andrà ad utilizzare il sistema.

Per poter utilizzare il sistema, l'utente dovrà essere registrato tramite username e password.

Per gestire la registrazione, il login e il logout è stato utilizzato il plugin SpringSecurity Service.

Come precondizione in fase di progettazione, è stata scelta come logica di funzionamento che gli utenti possiedano molte Board, cioè ne siano creatori e quindi amministratori ed inoltre che gli utenti possano avere molte card assegnate.



La classe board rappresenta l'oggetto principale del sistema.

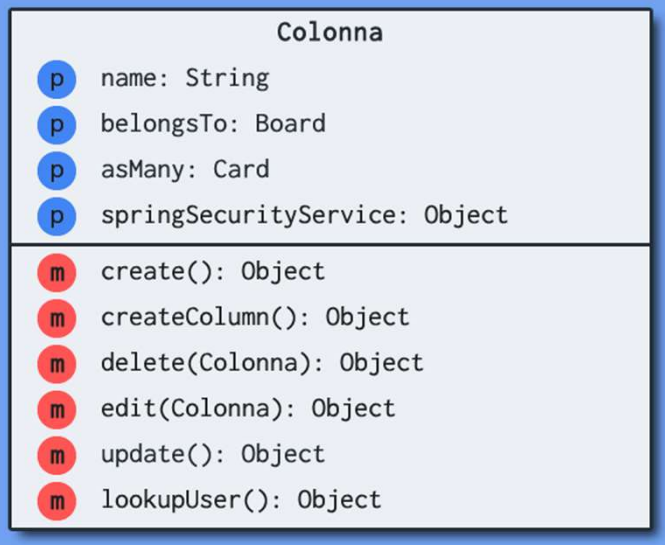
La board è stata rappresentata da un nome e da un Amministratore, cioè colui che l'ha creata.

Le precondizioni scelte in fase di progettazione della classe, hanno portato alla realizzazione di un modello che, ogni board contenga molte colonne e molti utenti intesi come partecipanti al progetto rappresentato dal nome della board, mentre è stato scelto che un solo utente sia l'amministratore.

La classe colonna rappresenta uno degli oggetti contenuti all'interno della board.

La colonna è rappresentata solo dal nome che gli viene assegnato in fase di creazione.

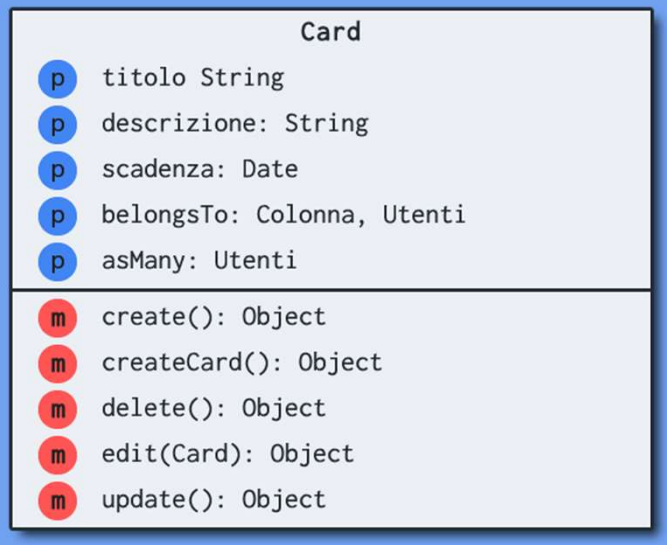
Progettato come un contenitore per gli oggetti di tipo card, in quanto ogni colonna può contenere più card, ed inoltre la colonna può essere contenuta solo in una board.



La classe card, rappresenta i task della kanban board.

Concepita con tre campi: il titolo che rappresenta in breve l'obbiettivo della card, la descrizione che descrive in dettaglio il task da compiere e una data di scadenza che rappresenta il tempo a disposizione per compiere il task.

Come precondizioni in fase di progettazione è stato deciso che ogni card deve appartenere ad una sola colonna



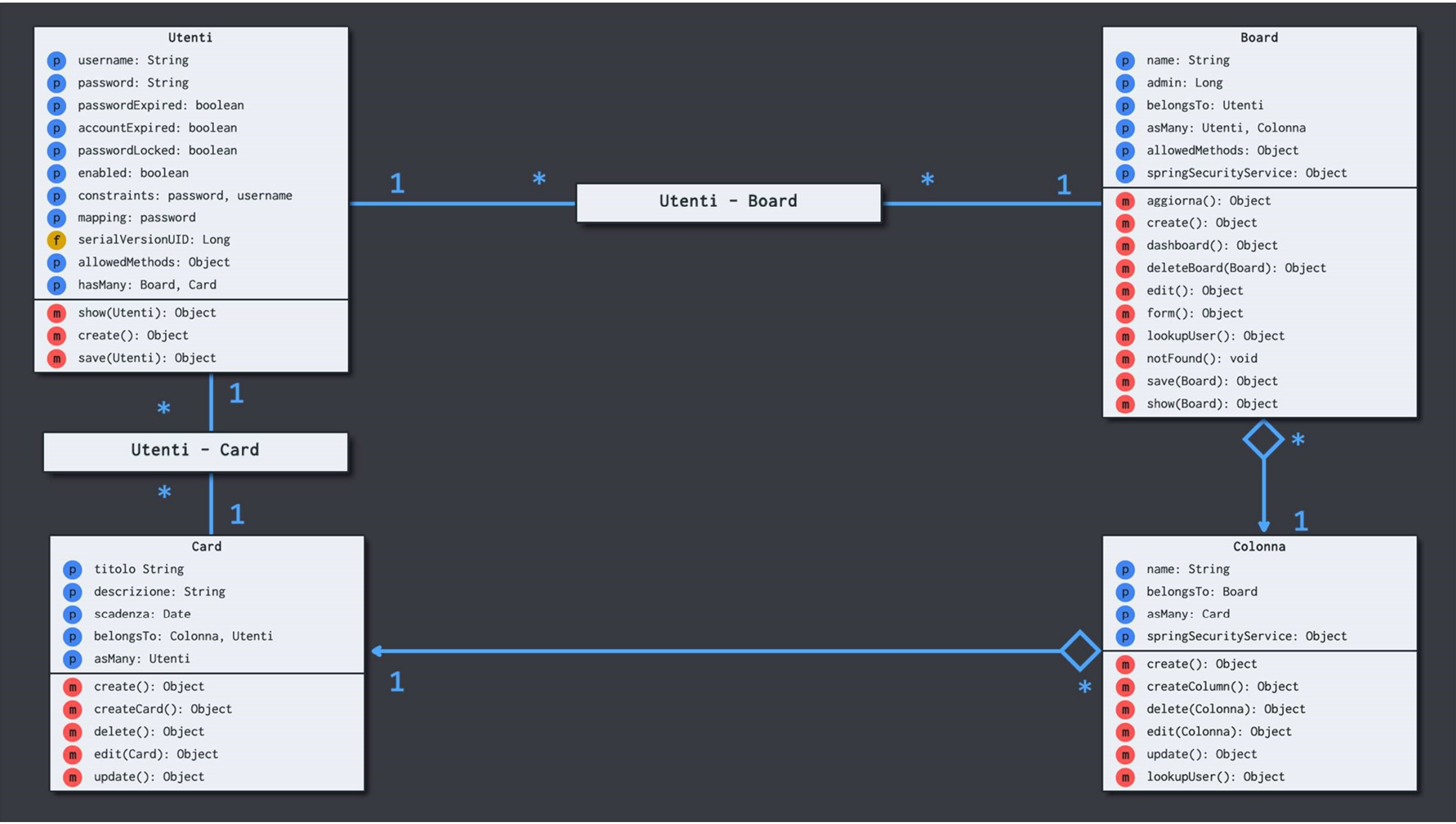
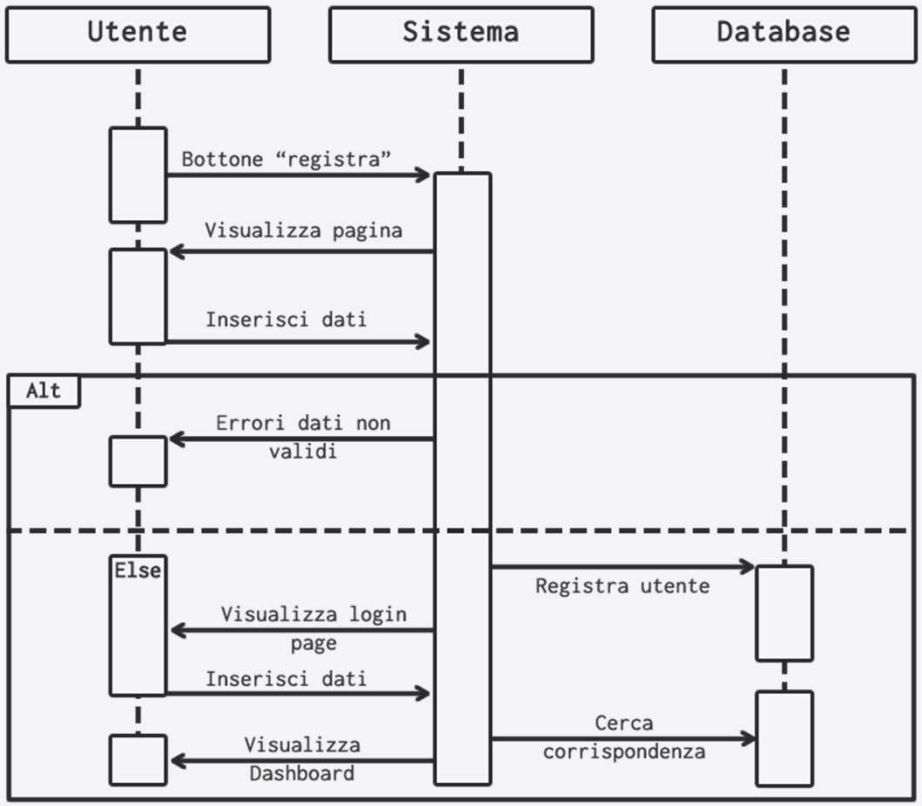

Utenti
username: String
password: String
passwordExpired: boolean
accountExpired: boolean
passwordLocked: boolean
enabled: boolean
constraints: password, username
mapping: password
serialVersionUID: Long
allowedMethods: Object
hasMany: Board, Card
show(Utenti): Object
create(): Object
save(Utenti): Object
1
*
Utenti - Board
*
1
Board
name: String
admin: Long
belongsTo: Utenti
asMany: Utenti, Colonna
allowedMethods: Object
springSecurityService: Object
aggiorna(): Object
create(): Object
dashboard(): Object
deleteBoard(Board): Object
edit(): Object
form(): Object
lookupUser(): Object
notFound(): void
save(Board): Object
show(Board): Object
*
1
Utenti - Card
*
1
Card
titolo String
descrizione: String
scadenza: Date
belongsTo: Colonna, Utenti
asMany: Utenti
create(): Object
createCard(): Object
delete(): Object
edit(Card): Object
update(): Object
1
*
*
1
Colonna
name: String
belongsTo: Board
asMany: Card
springSecurityService: Object
create(): Object
createColumn(): Object
delete(Colonna): Object
edit(Colonna): Object
update(): Object
lookupUser(): Object

Diagramma di sequenza:
Accesso al Sistema



Accesso al sistema

Test di accettazione: AccessoAlSistemaSpec
Priorità: -
Punti Storia: -

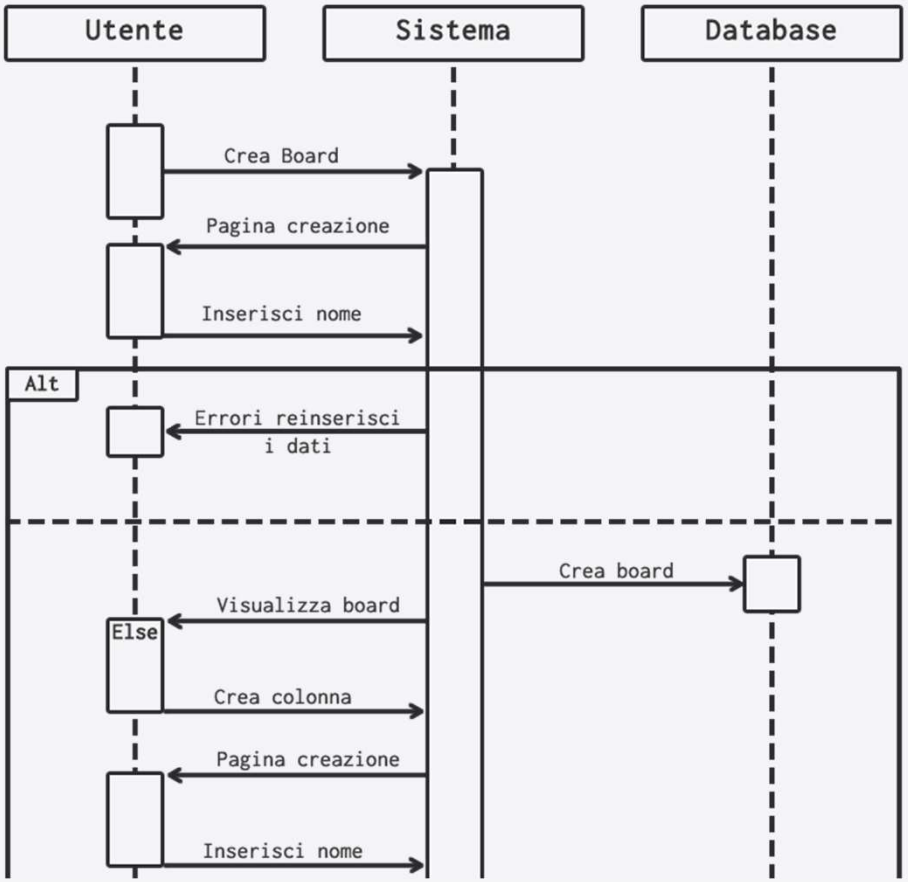
Un utente senza account vuole registrarsi al sistema per accedere alle funzionalità della KanbanBoard.

Sarà necessario premere su un bottone dedicato alla registrazione e inserire le credenziali che preferisce.

Se i dati sono considerati non validi, la pagina mostra un messaggio di errore sarà necessario reinserire i dati.

A registrazione completata si viene reindirizzati alla pagina di Login da cui reinserendo i propri dati si può accedere alla propria DashBoard

Diagramma di sequenza:
Creazione progetto (parte 1)



Creazione Progetto

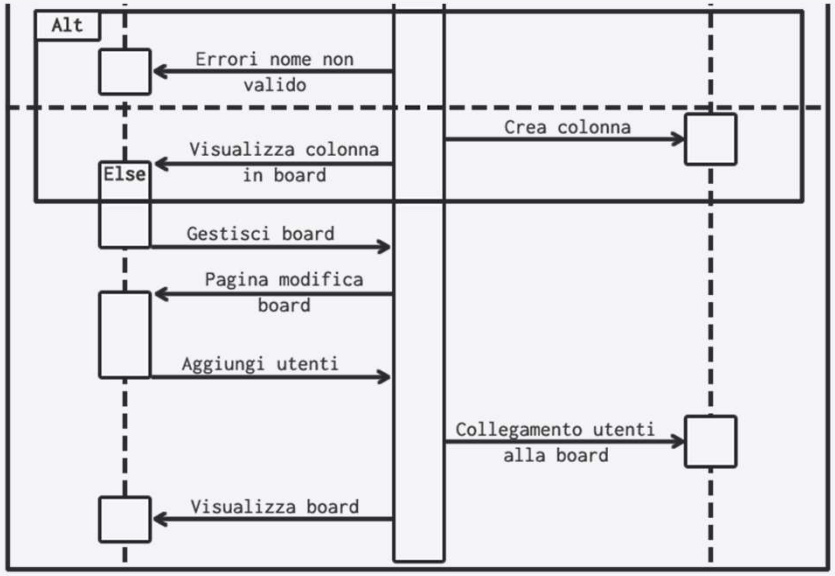
Test di accettazione: CreazioneProgettoSpec
Priorità: -
Punti Storia: -

Un utente registrato vuole creare una Board e le colonne di cui ha bisogno per gestire un progetto con due suoi colleghi: Davide e Giacomo, entrambi registrati nel sistema.

Se i nomi assegnati alla board e alle colonne sono validi, verrà mostrato una pagina con la propria board popolata dalle colonne, altrimenti i form di creazione mostreranno un messaggio di errore.

Una volta che l'utente ha creato la Board, la condivide con i suoi colleghi.

Diagramma di sequenza:
Creazione progetto (parte 2)



Creazione Progetto

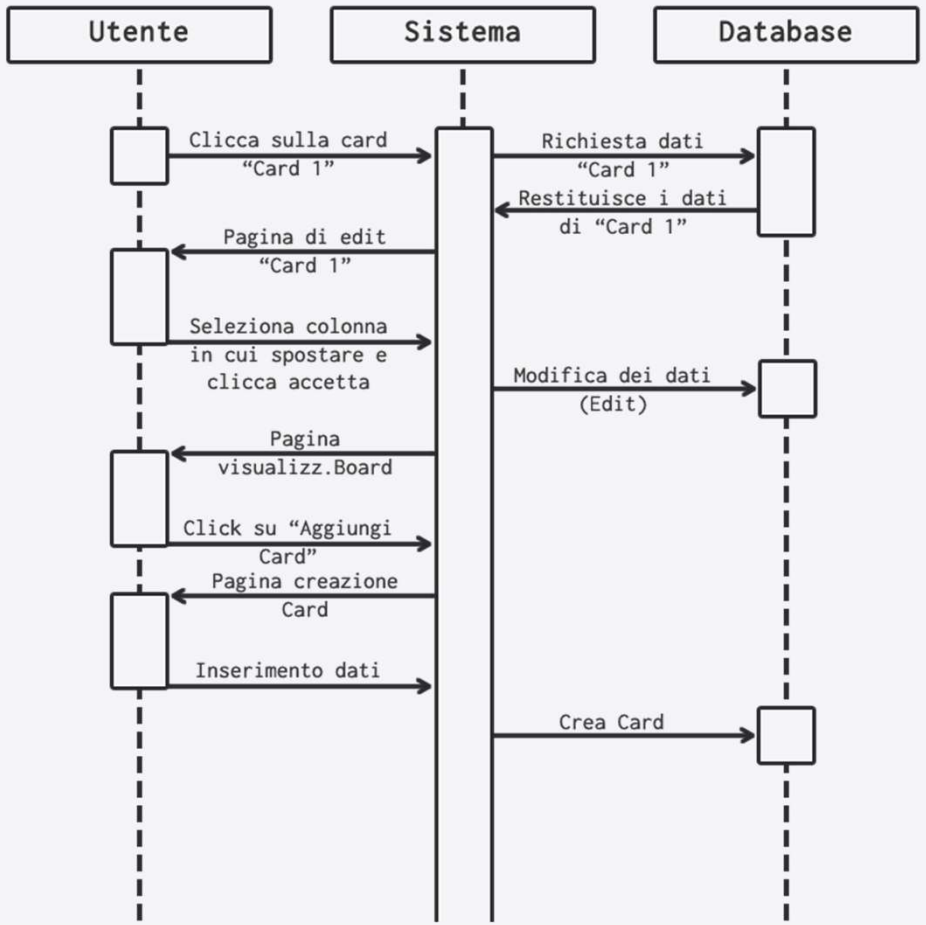
Test di accettazione: CreazioneProgettoSpec
Priorità: -
Punti Storia: -

Un utente registrato vuole creare una Board e le colonne di cui ha bisogno per gestire un progetto con due suoi colleghi: Davide e Giacomo, entrambi registrati nel sistema.

Se i nomi assegnati alla board e alle colonne sono validi, verrà mostrato una pagina con la propria board popolata dalle colonne, altrimenti i form di creazione mostreranno un messaggio di errore.

Una volta che l'utente ha creato la Board, la condivide con i suoi colleghi.

Diagramma di sequenza:
Gestione Card (parte 1)



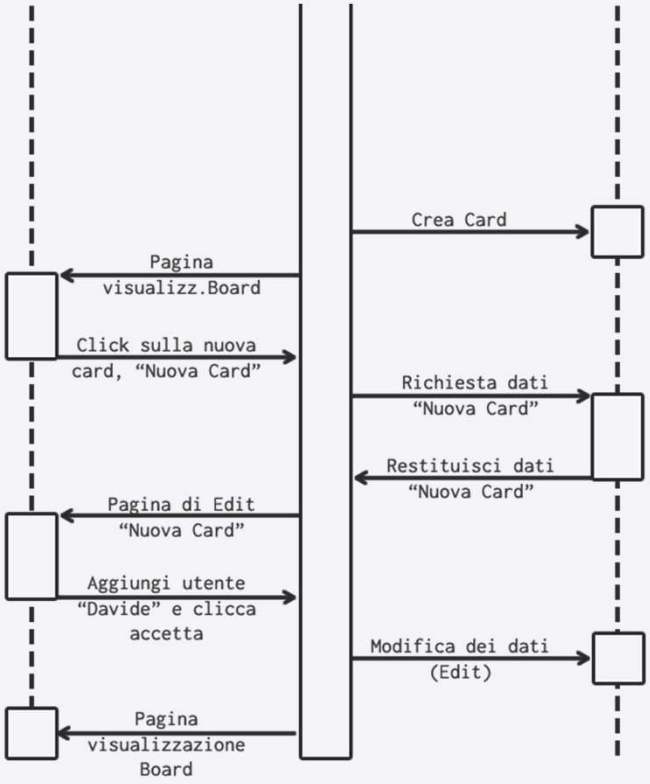
Gestione Card

Test di accettazione: GestioneCardSpec
Priorità: -
Punti Storia: -

Tre utenti che condividono la board "KanbanProject" hanno finito il task "Editing Colonna" e iniziato "Creazione Card".

Per cui devono spostare le relative card alla colonna successiva e crearne una nuova su "Backlog" da assegnare a uno degli utenti

Diagramma di sequenza:
Gestione Card (parte 2)



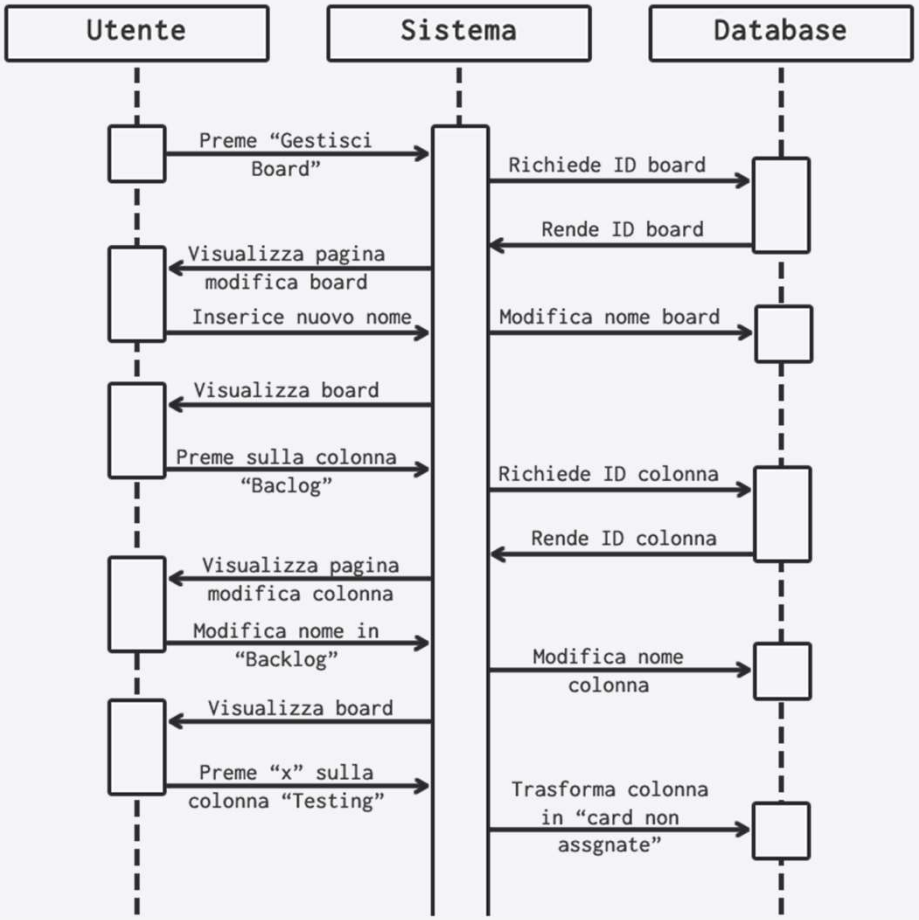
Gestione Card

Test di accettazione: GestioneCardSpec
Priorità: -
Punti Storia: -

Tre utenti che condividono la board “KanbanProject” hanno finito il task “Editing Colonna” e iniziato “Creazione Card”.

Per cui devono spostare le relative card alla colonna successiva e crearne una nuova su “Backlog” da assegnare a uno degli utenti

Diagramma di sequenza:
Modifica e cancellazione (parte 1)



Modifica e Cancellazione

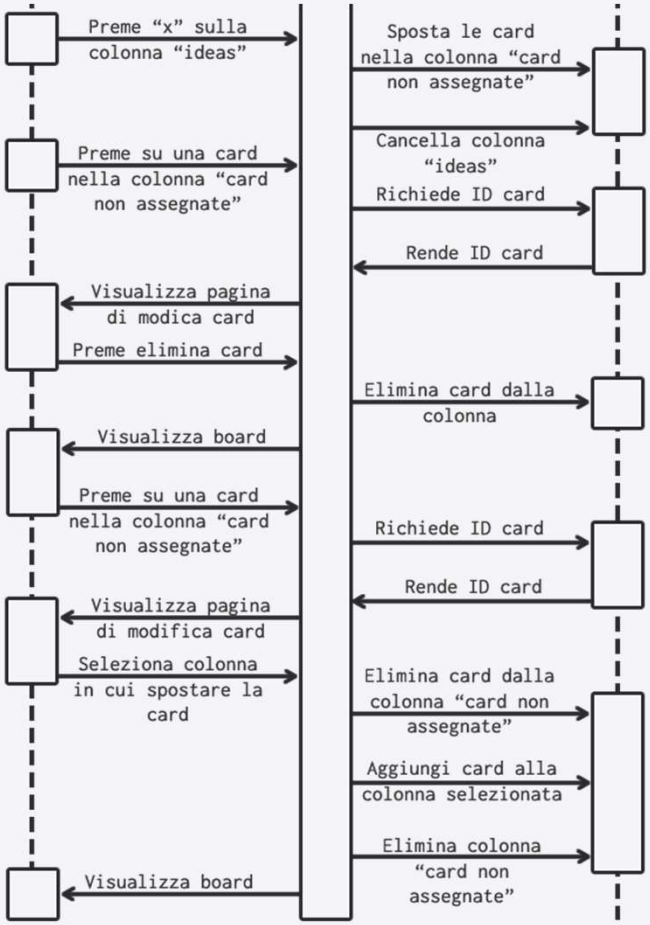
Test di accettazione: ModificaECancellazione
Priorità: -
Punti Storia: -

Un utente deve riprendere a distanza di tempo un progetto, e decide di modificare la Board del progetto che non ha mai ultimato per poter riutilizzare lo schema delle colonne senza doverle ricreare.
Ha però bisogno di rinominare il nome della board in “KanbanProject” e il titolo errato di una colonna.

Dopo aver riadattato la board, l’utente decide di eliminare due colonne che non gli servono: “Testing”, “Ideas”.

Le card al loro interno verranno spostate in una colonna speciale chiamata “Card non assegnate”. L’utente eliminerà quelle che non gli servono e sposterà su “Backlog” quelle che vuole tenere

Diagramma di sequenza:
Modifica e cancellazione (parte 2)



Modifica e Cancellazione

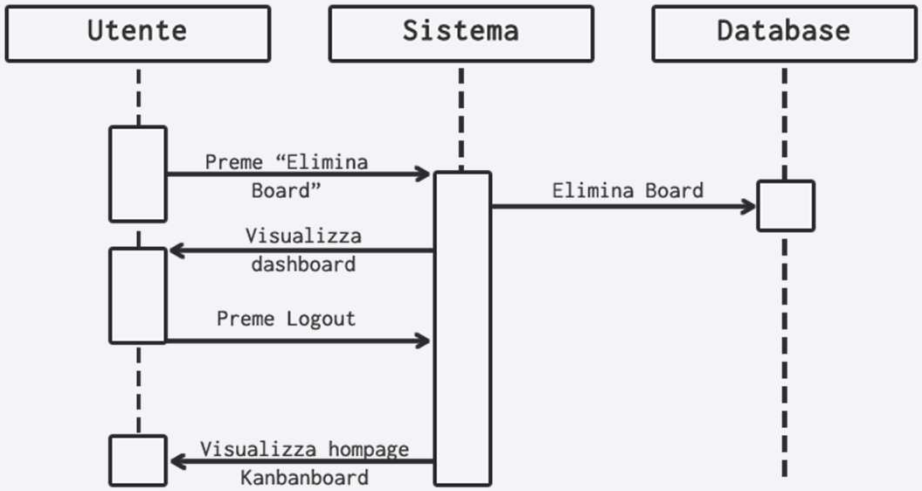
Test di accettazione: ModificaECancellazione
Priorità: -
Punti Storia: -

Un utente deve riprendere a distanza di tempo un progetto, e decide di modificare la Board del progetto che non ha mai ultimato per poter riutilizzare lo schema delle colonne senza doverle ricreare.
Ha però bisogno di rinominare il nome della board in “KanbanProject” e il titolo errato di una colonna.

Dopo aver riadattato la board, l’utente decide di eliminare due colonne che non gli servono: “Testing”, “Ideas”.

Le card al loro interno verranno spostate in una colonna speciale chiamata “Card non assegnate”. L’utente eliminerà quelle che non gli servono e sposterà su “Backlog” quelle che vuole tenere

Diagramma di sequenza:
Eliminazione Board e Logout



Eliminazione Board e Logout

Test di accettazione: EliminazioneBoardLogout
Priorità: -
Punti Storia: -

Un utente dopo aver passato molto tempo a sviluppare il suo prodotto, ha completato tutti i task necessari.

Decide quindi di eliminare la board per lasciare nella sua dashboard solo i progetti aperti, ed effettua poi il logout

ANALISI DEL SISTEMA

Il sistema al suo interno

Alcune parti fondamentali del codice interno al sistema

```
def create() {
    if (params.nomeBoard.equals(" ") || params.nomeBoard == null)
    {
        flash.message = "Il nome della board non può essere vuoto"
        redirect(controller: "Board" , action:"form")
    }
    else
    {
        def b = new Board(name: params.nomeBoard)
        b.setAdmin(lookupUser().id)
        b.addToUsers(lookupUser())
        b.save()
        lookupUser().save flush:true

        redirect(controller: "Board" , action:"show" , id: b.id)
    }
}
```

Creazione Board

Prendendo i dati dalla view inseriti da un utente, viene effettuato un controllo se i dati sono validi.

In caso di dati non idonei viene mostrato un messaggio di errore e si viene reindirizzati alla pagina di creazione. Altrimenti se validi la Board viene creata con il nome scelto dall'utente, ed inoltre l'utente viene impostato come amministratore. Si viene poi reindirizzati alla Board appena creata.

```
def aggiorna() {

    def idBoard = params.idBoard

    if (params.nome == null || params.nome.equals(" ")) {
        flash.message = "Il campo 'nome' non può essere vuoto"
        redirect(controller: "Board" , action: "edit" , id: idBoard)
    } else {

        Board.findById(idBoard).name = params.nome
        Board.findById(idBoard).save(flush:true)

        if (params.utenteSelezionato != null) {
            (params.utenteSelezionato).each {
                Board.findById(idBoard).addToUsers(Utenti.findById(it))
                Utenti.findById(it).addToBoards(Board.findById(idBoard))
                Board.findById(idBoard).save(flush: true)
                Utenti.findById(it).save(flush: true)
            }
        }
        if (params.utenteSelezionatoRimuovi != null) {
            (params.utenteSelezionatoRimuovi).each {
                Board.findById(idBoard).removeFromUsers(Utenti.findById(it))
                Utenti.findById(it).removeFromBoards(Board.findById(idBoard))
                Board.findById(idBoard).save(flush: true)
                Utenti.findById(it).save(flush: true)
            }
        }
        redirect(controller: "Board", action: "show", id: idBoard)
    }
}
```

Aggiornamento dati Board

Prendendo i dati dalla view di modifica inseriti dall'utente, viene effettuato un controllo se i dati sono validi.

In caso di dati non idonei viene mostrato un messaggio di errore e si viene reindirizzati alla pagina modifica. Altrimenti vengono effettuate le modifiche.

Aggiorna() si occupa di inserire e/o eliminare utenti partecipanti alla board oltre a consentire la modifica del nome. A modifiche effettuate si viene reindirizzati alla board.

```
def deleteBoard(){

    def idBoard = session.currentBoard

    Board.findById(idBoard).users.collect().each {
        Board.findById(idBoard).removeFromUsers(it)
    }

    Utenti.findById(lookupUser().id).save(flush:true)
    Board.findById(idBoard).delete(flush: true)

    redirect(controller: "Board" , action:"dashboard")

}
```

Eliminazione Board

Dopo la pressione da parte dell'utente del tasto di eliminazione, dalla sessione corrente viene recuperato l'id della board corrente (session.currentBoard).

In sequenza vengono eliminati tutti gli utenti che partecipano alla Board e per ricorsione anche le colonne e le card.

Dopo di che si viene reindirizzati alla dashboard.

```
def createColumn(){
    if (params.nomeColonna == null ||
        params.nomeColonna.equals(" ") ||
        params.nomeColonna.equals("Card non assegnate")) {

        if(params.nomeColonna.equals("Card non assegnate")){
            flash.message = "La colonna non può chiamarsi 'Card non assegnate' "
            redirect(controller: "Colonna" , action:"create")
        }
        else{
            flash.message = "Il campo 'Nome della colonna' non può essere vuoto"
            redirect(controller: "Colonna" , action:"create")
        }
    }
    else {
        def c = new Colonna(name: params.nomeColonna)
        Board.findById(session.currentBoard).addToColonne(c)
        Board.findById(session.currentBoard).save(flush: true)
        c.save(flush:true)
        redirect(controller: "Board" , action:"show" ,
                 id: Board.findById(session.currentBoard).id)
    }
}
```

Creazione Colonna

Prelevando i dati inseriti nella view dall'utente, viene effettuato un controllo se sono validi.

Se non sono validi, si viene reindirizzati alla pagina di creazione con un messaggio di errore.

Caso particolare di dati non validi è il caso in cui, l'utente voglia assegnare il nome 'Card non assegnate', in quanto riservato del sistema, in caso di eliminazione di colonne con all'interno delle card.

In caso di dati validi, la colonna viene creata e associata alla board corrente. Dopo di che si viene reindirizzati alla view della board.

```
def update() {

    def idColonna = params.idColonna
    def idUser = lookupUser().id
    session.currentUser = lookupUser().id

    if (params.nomeColonna == null ||
        params.nomeColonna.equals(" ") ||
        params.nomeColonna.equals("Card non assegnate")) {

        if(params.nomeColonna.equals("Card non assegnate")){
            flash.message = "La colonna non può chiamarsi 'Card non assegnate' "
            redirect(controller: "Colonna" , action:"edit" ,
                     id: Colonna.findById(session.currentCol).id ,
                     params: [idCNA: params.idCNA])
        }
        else{
            flash.message = "Il campo 'Nome' non può essere vuoto"
        redirect(controller: "Colonna" , action:"edit" ,
                 id: Colonna.findById(session.currentCol).id ,
                 params: [idCNA: params.idCNA])
        }
    }
    else {
        Colonna.findById(idColonna).name = params.nomeColonna
        Colonna.findById(idColonna).save(flush:true)

        if(params.currentCards != null) {
            if(params.currentCards instanceof String) {
                Utenti.findById(idUser).removeFromCards(Card.findById(params.currentCards))
                Colonna.findById(idColonna).removeFromCards(Card.findById(params.currentCards))
                Card.findById(params.currentCards).removeFromUsers(Utenti.findById(idUser))

                Utenti.findById(idUser).save(flush: true)
                Colonna.findById(idColonna).save(flush: true)
                Card.findById(params.currentCards).delete(flush: true)
            } else {
                (params.currentCards).each {
                    Utenti.findById(idUser).removeFromCards(Card.findById(it))
                    Colonna.findById(idColonna).removeFromCards(Card.findById(it))
                    Card.findById(it).removeFromUsers(Utenti.findById(idUser))

                    Utenti.findById(idUser).save(flush: true)
                    Colonna.findById(idColonna).save(flush: true)
                    Card.findById(it).delete(flush: true)
                }
            }
```

Aggiornamento dati Colonna (parte 1)

Prelevando i dati dalla view di modifica inseriti dall'utente, viene effettuato un controllo se sono validi.

In caso di dati non validi, viene mostrato un messaggio di errore e si viene reindirizzati alla view di modifica.

Nel caso i dati siano validi la colonna viene salvata.

Nella funzione update è possibile gestire anche le card della colonna, nello specifico cancellare una o più card.

```
        }

        if(params.CNACards != null) {
            if(params.CNACards instanceof String) {
                Colonna.findById(params.idCNA).removeFromCards(Card.findById(params.CNACards))
                Colonna.findById(idColonna).addToCards(Card.findById(params.CNACards))
                Colonna.findById(params.idCNA).save(flush:true)
                Colonna.findById(idColonna).save(flush:true)

                if(Colonna.findById(params.idCNA).cards.size() == 0)
                {
                    Board.findById(session.currentBoard).removeFromColonne(Colonna.findById(params.idCNA))
                    Board.findById(session.currentBoard).save(flush:true)
                    Colonna.findById(params.idCNA).delete(flush:true)
                }
            } else {
                (params.CNACards).each {
                    Colonna.findById(params.idCNA).removeFromCards(Card.findById(it))
                    Colonna.findById(idColonna).addToCards(Card.findById(it))
                    Colonna.findById(params.idCNA).save(flush:true)
                    Colonna.findById(idColonna).save(flush:true)

                    if(Colonna.findById(params.idCNA).cards.size() == 0)
                    {
                        Board.findById(session.currentBoard).removeFromColonne(Colonna.findById(params.idCNA))
                        Board.findById(session.currentBoard).save(flush:true)
                        Colonna.findById(params.idCNA).delete(flush:true)
                    }
                }
            }
        }

        redirect(controller: "Board" , action:"show" , id: Board.findById(session.currentBoard).id )
    }
}
```

Aggiornamento dati Colonna (parte 2)

Nella funzione update(), viene gestito lo spostamento delle eventuali card dalla colonna ‘Card non assegnate’.

Questa colonna riservata, viene visualizzata quando vi sono card orfane di colonne e viene automaticamente cancellata quando è priva di card.

Aggiornati i dati, si viene reindirizzati alla view della Board.

```
def delete(Colonna) {

    if (colonna == null) {
        colonna = Colonna.findById(session.currentCol)
    }

    def hasCardNonAssegnate = false
    def cardNonAssegnateID
    def cardsCount = 0
    Board.findById(session.currentBoard).colonne.each {
        if (it.name.equals("Card non assegnate")){
            hasCardNonAssegnate = true
            cardNonAssegnateID = it.id
        }
    }
    colonna.cards.each {cardsCount++}

    if (hasCardNonAssegnate == true){
        Colonna.findById(cardNonAssegnateID).cards.addAll(colonna.cards)
        Board.findById(session.currentBoard).removeFromColonne(colonna)
        colonna.cards.removeAll(colonna.cards)
        colonna.delete flush:true
    }
    else {
        if (cardsCount > 0){
            colonna.name = "Card non assegnate"
            colonna.save flush:true
        } else{
            Board.findById(session.currentBoard).removeFromColonne(colonna)
            colonna.delete flush:true
        }
    }
    redirect(controller: "Board" , action:"show" ,
            id: Board.findById(session.currentBoard).id)
}
```

Eliminazione Colonna

Quando l'utente seleziona l'eliminazione della colonna, viene recuperato l'id della colonna scelta per l'eliminazione.

Se la colonna che si vuole eliminare possiede delle card, esse diventano orfane e vengono spostate nella colonna 'Card non assegnate' ed in caso essa non esista, viene creata. In caso la colonna non possied card, il passaggio delle card viene saltato.

Eliminata la colonna si viene reindirizzati alla Board.

```
def createCard(){
    if(params.nomeCard.equals(" ") || params.descrizione.equals(" ")) {
        flash.message = "I campi non possono essere vuoti"
        redirect(controller: "Card" , action:"create")
    } else {
        def card = new Card(titolo: params.nomeCard,
                            descrizione: params.descrizione,
                            scadenza: params.scadenza)

        Colonna.findById(session.currentCol).addToCards(card)
        Colonna.findById(session.currentCol).save(flush: true)
        card.save(flush: true)
        redirect(controller: "Board", action: "show",
                 id: Board.findById(session.currentBoard).id)
    }
}
```

Creazione Card

Quando l'utente dalla schermata di creazione della card da l'ok per la creazione, viene richiamata la funzione createCard(), che recuperando i dati inseriti dall'utente nella view, ne esegue il controllo e se validi crea la card, altrimenti viene visualizzato un messaggio di errore e si viene reindirizzati alla pagina di creazione.

Parte fondamentale della creazione della card è stata la progettazione della vista, in quanto il pulsante di creazione è innestato all'interno di ogni singola colonna. Questo ha permesso di non doversi occupare della gestione della colonna per il salvataggio della card.

```
def update() {

    def idCard = params.idCard
    def idColonna = session.currentCol

    if ( params.titolo.equals(" ") || params.titolo == null ||
       |params.descrizione.equals(" ") || params.descrizione == null) {
        flash.message = "I campi non possono essere vuoti"
        render(view: "edit", model: [currentCard: Card.findById(idCard) ,
                                     currentColumn: Colonna.findById(idColonna) ,
                                     currentBoard: Board.findById(session.currentBoard)])
    }
    else {
        Card.findById(idCard).titolo = params.titolo
        Card.findById(idCard).descrizione = params.descrizione
        Card.findById(idCard).scadenza = params.scadenza

        if (params.utentiBoard!=null) {
            (params.utentiBoard).each {
                Card.findById(idCard).addToUsers(Utenti.findById(it))
                Utenti.findById(it).addToCards(Card.findById(idCard))
                Utenti.findById(it).save(flush: true)
            }
        }

        if (params.utentiCard!=null) {
            (params.utentiCard).each {
                Card.findById(idCard).removeFromUsers(Utenti.findById(it))
                Utenti.findById(it).removeFromCards(Card.findById(idCard))
                Utenti.findById(it).save(flush: true)
            }
        }

        if (params.colonnaCard!=null && params.colonnaCard != ('null')) {
            Colonna.findById(session.currentCol).removeFromCards(Card.findById(idCard))
            Colonna.findById(params.colonnaCard).addToCards(Card.findById(idCard))
        }

        if(idColonna != null && Colonna.findById(idColonna).cards.size() == 0 &&
           Colonna.findById(idColonna).name.equals("Card non assegnate") ) {
            Board.findById(session.currentBoard).removeFromColonne(Colonna.findById(idColonna))
            Board.findById(session.currentBoard).save(flush:true)
            Colonna.findById(idColonna).delete(flush:true)
        }
        Card.findById(idCard).save(flush:true)
        redirect(controller: "Board" , action:"show" ,
                 id: Board.findById(session.currentBoard).id )
    }
}
```

Aggiornamento dati Card

Dalla view di modifica della card viene richiamata la funzione update() che si occupa di aggiornare i dati della card selezionata.

Effettuato il controllo sulla validità dei dati, se non validi viene visualizzato un messaggio di errore e si viene reindirizzati alla view di modifica dei dati. Altrimenti i dati vengono salvati.

Dalla funzione update() sono gestibili anche gli utenti a cui è assegnata la card, prelevandoli esclusivamente dai partecipanti alla Board. Inoltre è possibile spotare una card da una colonna ad un’altra, della Board presa in esame.

```
def delete(){

    def idColonna = params.currentCol
    def idCard = params.currentCard
    def idCNA

    if (params.idCNA != null){idCNA = params.idCNA}

    Card.findById(idCard).users.collect().each {
        Card.findById(idCard).removeFromUsers(it)
    }
    Card.findById(idCard).save(flush:true)
    Colonna.findById(idColonna).removeFromCards(Card.findById(idCard))
    Colonna.findById(idColonna).save(flush:true)
    Card.findById(idCard).delete(flush:true)

    if(idCNA != null && Colonna.findById(idCNA).cards.size() == 0 &&
       Colonna.findById(idColonna).name.equals("Card non assegnate") ) {

        Board.findById(session.currentBoard).removeFromColonne(Colonna.findById(idCNA))
        Board.findById(session.currentBoard).save(flush:true)
        Colonna.findById(idCNA).delete(flush:true)
    }
    redirect(controller: "Board" , action:"show" ,
             id: Board.findById(session.currentBoard).id )
}
```

Eliminazione Card

Alla pressione del tasto di eliminazionde della card, viene invocata la funzione delete(). La funzione recupera l'id della card corrente. Se la card è l'ultima presente in 'Card non assegnate' la colonna viene automaticamente eliminata.

Eliminata la card si viene reindirizzati alla board.

```
def save(Utenti utenti){
    if (utenti == null) {
        transactionStatus.setRollbackOnly()
        notFound()
        return
    }
    if (utenti.hasErrors()) {
        transactionStatus.setRollbackOnly()
        if(Utenti.findByUsername(params.username) != null &&
           Utenti.findByUsername(params.username).count() > 0 ||
           params.username == " " || params.username == "") {
            if (params.username == " " || params.username == ""){
                flash.message = "Credenziali non valide"
            }else{
                flash.message = "Questo username esiste già"
            }
        }
        respond utenti.errors, view:'create'
        return
    }
    utenti.save(flush: true)
    request.withFormat {
        form multipartForm {
            flash.message = message(code: 'default.created.message',
                                    args: [message(code: 'utenti.label',
                                    default: 'Utenti'), utenti.id])
            redirect utenti
        }
        '*' { respond utenti, [status: CREATED] }
    }
}
```

Creazione utenti

Dalla view di creazione dell'utente, viene richiamata la funzione save() che effettua un controllo sulle credenziali inserite e si assicura che non siano già presenti nel database. In caso di username già presente, si viene reindirizzati alla view di creazione con un messaggio di errore, altrimenti viene salvato con successo l'utente.

TEST SUL SISTEMA

Test unitari sul sistema

Alcuni test effettuati sui domini e sui controller del sistema.

TEST SISTEMA

Board: Test sul controller

```
void "Show: creazione di una Board completa"(){
    when: "La creazione è stata eseguita"
        board(params)
        def b = new Board(params)
        user(params)
        def u = new Utenti(params)
        b.admin = u.id
        controller.show(b)

    then:"Il modello non è nullo, quindi è creato"
        model.board != null
}
```

Creata una generica board, il modello vine correttamente creato

```
void "Aggiorna: test creazione modello corretto"(){
    when:"Invocazione del metodo aggiorna"
        board(params)
        def b = new Board(params)
        user(params)
        def u = new Utenti(params)
        b.admin = u.id
        controller.aggiorna()

    then:"Il modello è correttamente visualizzato"
        model.board != null
}
```

Data una generica board, l'invocazione del metodo aggiorna crea il modello corretto

TEST SISTEMA

Board: Test sul dominio

```
void "Creazione Board"() {
    given:"Creo board"
    new Board(name: "Board" , admin: 1).save(flush:true)

    expect:"La board esiste"
        Board.findAll().size()==1
}

void "Rinomina Board"() {
    given:"Creo board e rinomino"
    new Board(name: "Board" , admin: 1).save(flush:true)
    Board.findByName("Board").name = "Board2"

    expect:"La board esiste con un altro nome "
    Board.findByAdmin(1).name == "Board2"
}

void "Ricerca per Admin"() {
    given:"Creo le board"
    new Board(name: "B1" , admin: 1).save(flush:true)
    new Board(name: "B2" , admin: 3).save(flush:true)

    expect:"La board viene trovata"
    Board.findByAdmin(1).size() == 1
}

void "Creo la board e aggiungo una colonna"() {
    given:"Creo una colonna e la aggiungo alla board"
    def board = new Board(name: "Board" , admin: 1).save(flush:true)
    def colonna = new Colonna(name: "colonna").save(flush: true)
    board.addToColonne(colonna).save(flush: true)

    expect:"La colonna viene trovata"
    Board.findByAdmin(1).colonne.count(colonna) == 1
}
```

Conteggio numero di board dopo una singola creazione

Rinomiare una generica board

Ricerca tra più board via amministratore

Creo una board pronta all’uso

TEST SISTEMA

Board: Test sul dominio

```
void "Creo la board e aggiungo due utenti"() {
    given:"Creo la board, creo due utenti e li aggiungo alla board"
    def board = new Board(name: "Board" , admin: 1).save(flush:true)
    def u1 = new Utenti(username: "u1", password: "u1").save(flush: true)
    def u2 = new Utenti(username: "u2", password: "u2").save(flush: true)
    board.addToUsers(u1)
    board.addToUsers(u2)

    expect:"Gli utenti sono presenti"
    Board.findByAdmin(1).users.size() == 2
}

void "Ricerca tra board multiple"() {
    given:"Creo 4 board"
    new Board(name: "Board 1" , admin: 1).save(flush:true)
    new Board(name: "Board 2" , admin: 11).save(flush:true)
    new Board(name: "Board 3" , admin: 21).save(flush:true)
    new Board(name: "Board 4" , admin: 31).save(flush:true)

    expect:"La board viene trovata"
    Board.findByName("Board 4") != null
}

void "Numero di board di un utente"() {
    given:"Creo 4 board"
    new Board(name: "Board 1" , admin: 1).save(flush:true)
    new Board(name: "Board 2" , admin: 1).save(flush:true)
    new Board(name: "Board 3" , admin: 1).save(flush:true)
    new Board(name: "Board 4" , admin: 1).save(flush:true)

    expect:"L'utente è admin di 4 board"
    Board.findByAdmin(1).count() == 4
}
```

Creata una generica board, aggiungo due utenti al progetto

Creato un numero arbitrario di board, ricerco la board tramite il suo nome

Conteggio numero di board create da un singolo utente

TEST SISTEMA

Colonna: Test sul controller

```
void "Test sulla create se rende il modello corretto"() {
    when: "The create action is executed"
    populateValidParams(params)
    def col = new Colonna(params)
    controller.createColumn()

    then: "The model is correctly created"
    model.colonna != null
}

void "Test salvatagggio nuova colonna"() {
    when: "Viene creata una nuova colonna"
    populateValidParams(params)
    def col = new Colonna(params)
    controller.createColumn()

    then: "l'istanza viene salvata"
    col != null
}
```

Creata una generica colonna, il modello vine correttamente creato

Creata una generica colonna, la colonna esiste

TEST SISTEMA

Colonna: Test sul dominio

```
void "Rinomina colonna"() {
    given:"Creo la colonna e rinomino"
    def col = new Colonna(name: "colonna").save(flush:true)
    Colonna.findByName("colonna").name = "colonna2"

    expect:"La colonna esiste con un altro nome "
    Colonna.findById(col.id).name == "colonna2"
}

void "Ricerca per ID colonna"() {
    given:"Creo la colonna e salvo l'id"
    def col = new Colonna(name: "colonna").save(flush:true)
    def id = col.id

    expect:"La colonna viene trovata"
    Colonna.findById(id).id == col.id
}

void "Inserimento più colonne in una board"() {
    given: "Creo la colonna e salvo l'id"
    def board = new Board(name: "Board", admin: 1).save(flush: true)
    def c1 = new Colonna(name: "c1").save(flush: true)
    def c2 = new Colonna(name: "c2").save(flush: true)
    def c3 = new Colonna(name: "c3").save(flush: true)
    board.addToColonne(c1).save(flush: true)
    board.addToColonne(c2).save(flush: true)
    board.addToColonne(c3).save(flush: true)

    expect: "Le colonne sono state inserite tutte"
    board.colonne.size() == 3
}
```

Data una colonna presistente, cambio il nome

Ricerca tra le colonne esistenti via id della colonna

Inserimento multiplo di colonne in una board

TEST SISTEMA

Colonna: Test sul dominio

```
void "Creo board, colonna e card e li lego tramite le dipendenze"(){
    given:"Creo una colonna e la aggiungo alla board"
    def board = new Board(name: "Board" , admin: 1).save(flush:true)
    def colonna = new Colonna(name: "colonna").save(flush: true)
    def card = new Card(titolo: "card",
                        descrizione: "descrzione",
                        scadenza: new Date('10/10/2020')).save(flush: true)
    board.addToColonne(colonna).save(flush: true)
    colonna.addToCards(card).save(flush: true)
    def id = colonna.id

    expect:"Tutto viene legato per dipendenze"
    Board.findByAdmin(1).colonne.count(colonna) == 1
    Colonna.findById(id).cards.count(card) == 1
}

void "Trovare colonne all'interno di una board"(){
    given:"Creo una colonna e la aggiungo alla board"
    def b1 = new Board(name: "b1" , admin: 1).save(flush:true)
    def b2 = new Board(name: "b2" , admin: 1).save(flush:true)
    def c1 = new Colonna(name: "c1").save(flush: true)
    def c2 = new Colonna(name: "c2").save(flush: true)
    b1.addToColonne(c1).save(flush: true)
    b2.addToColonne(c2).save(flush: true)

    expect:"le colonne vengono trovate dentro la board a cui sono state assegnate"
    Board.findByName(b1.name).getColonne().name.toString() ==
        "[" + Colonna.findById(c1.id).getName().toString() + "]"
    Board.findByName(b2.name).getColonne().name.toString() ==
        "[" + Colonna.findById(c2.id).getName().toString() + "]"
}
```

Creazione di una board, con una colonna e una card

Ricerca colonne tramite il nome della board

TEST SISTEMA

Card: Test sul controller

```
void "Show: creazione di una Card completa"(){
    when: "La creazione è stata eseguita"
    populateValidParams(params)
    def card = new Card(params)
    user(params)
    def user = new Utenti(params)
    lu.add(user)
    card.users = lu
    colonn(params)
    def col = new Colonna(params)
    lc.add(card)
    col.cards = lc

    then: "Il modello è correttamente creato"
    !model.card
}

void "Controllo creazione card su tutti i parametri"(){
    when: "La creazione è stata eseguita"
    populateValidParams(params)
    def card = new Card(params)
    user(params)
    def user = new Utenti(params)
    lu.add(user)
    card.users = lu
    colonn(params)
    def col = new Colonna(params)
    lc.add(card)
    col.cards = lc


    then: "La card è completa"
    card.titolo != null
    card.descrizione != null
    card.users.size() == 1
}
```

Creando una generica card, il modello viene creato

Creata una generica card, i sui campi non sono mai nulli

TEST SISTEMA

Card: Test sul dominio

```
void "Cambio la scadenza card"() {
    given:"Creo la card e cambio la scadenza"
    def card = new Card(titolo: "card",
                        descrizione: "descrzione",
                        scadenza: new Date('10/10/2020')).save(flush: true)
    Card.findById(card.id).scadenza = new Date('8/8/2020')

    expect:"La colonna esiste con un altro nome "
    Card.findById(card.id).scadenza == new Date('8/8/2020')
}

void "Ricerca per ID crad"() {
    given:"Creo la card e salvo l'id"
    def card = new Card(titolo: "card",
                        descrizione: "descrzione",
                        scadenza: new Date('10/10/2020')).save(flush: true)
    def id = card.id

    expect:"La colonna viene trovata"
    Card.findById(id).id == card.id
}

void "Creo board, colonna e card e li lego tramite le dipendenze"() {
    given:"Creo una colonna e la aggiungo alla board"
    def board = new Board(name: "Board" , admin: 1).save(flush:true)
    def colonna = new Colonna(name: "colonna").save(flush: true)
    def card = new Card(titolo: "card",
                        descrizione: "descrzione",
                        scadenza: new Date('10/10/2020')).save(flush: true)
    board.addToColonne(colonna).save(flush: true)
    colonna.addToCards(card).save(flush: true)
    def id = colonna.id
    def id2 = card.id

    expect:"Tutto viene legato per dipendenze"
    Board.findByAdmin(1).colonne.count(colonna) == 1
    Colonna.findById(id).cards.count(card) == 1
    Card.findById(id2).id == card.id
}
```

Modifica campo scadenza della card

Ricerca card tramite id

Creazione di una board, con una colonna e una card con verifica anche su card

TEST SISTEMA

Utenti: Test sul controller

```
void "Test salvataggio nuovo utente"() {
    when: "viene eseguita l'azione saveNewUser con un'istanza valida"
    populateValidParams(params)
    def user = new Utenti(params)

    then: "l'istanza viene salvata"
    user != null
}
```

Salvataggio nuovo utente

TEST SISTEMA

Utenti: Test sul dominio

```
void "Creazione Utente"() {
    given:"Creo un utente"
    def user = new Utenti(username: "utente",
                          password: "1234").save(flush: true)

    expect:"L'utente esiste"
    Utenti.findAll().size()==1
}

void "Cambio nome utente"() {
    given:"Creo l'utente e cambio l'username"
    def user = new Utenti(username: "utente",
                          password: "1234").save(flush: true)
    Utenti.findByUsername("utente").username = "utente2"

    expect:"Nome utente cambiato"
    Utenti.findById(user.id).username == "utente2"
}

void "Cambio password utente"() {
    given:"Creo l'utente e cambio la password"
    def user = new Utenti(username: "utente",
                          password: "1234").save(flush: true)
    Utenti.findById(user.id).password = "5678"

    expect:"Password cambiata"
    Utenti.findById(user.id).password == "5678"
}
```

Creazione nuovo utente tramite parametri della view

Cambio valore dell’username utente

Cambio valore password dell’utente

TEST SISTEMA

Utenti: Test sul dominio

```
void "Creo utente che poi crea una board"() {
    given:"Creo l'utente, poi la board"
    def user = new Utenti(username: "utente", password: "1234").save(flush: true)
    def board = new Board(name: "Board" , admin: user.id).save(flush:true)

    expect:"L'admin della board è user"
    board.admin == user.id
}
```

```
void "Creo utente e gli assegno una card"() {
    given:"Creo l'utente, poi la board"
    def user = new Utenti(username: "utente", password: "1234").save(flush: true)
    def card = new Card(titolo: "card",
                        descrizione: "descrizione",
                        scadenza: new Date('10/10/2020')).save(flush: true)
    card.addToUsers(user).save(flush: true)

    expect:"La card è stata assegnata"
    Card.findById(card.id).users.size() == 1
}
```

```
void "Tentativo di registrare due utenti con lo stesso nome"(){
    given:"Creo due utenti con lo stesso nome"
    def user1 = new Utenti(username: "utente", password: "1234").save(flush: true)
    def user2 = new Utenti(username: "utente", password: "5678").save(flush: true)

    expect:"user2 non deve esistere"
    user2 == null
}
```

Creazione utente, che crea una board che ne diventa amministratore

Creo una card e un utente e assegno la card a quest'ultimo

Tentativo di registrare un utente con credenziali già utilizzate

TEST SUL SISTEMA

Test di Accettazione e User Stories

Alcuni test che simulano le azioni che un utente del sistema può compiere durante l’utilizzo

TEST DI INTEGRAZIONE

Accesso al Sistema

Accesso al sistema

Test di accettazione: AccessoAlSistemaSpec
Priorità: -
Punti Storia: -

Un utente senza account vuole registrarsi al sistema per accedere alle funzionalità della KanbanBoard.

Sarà necessario premere su un bottone dedicato alla registrazione e inserire le credenziali che preferisce.

Se i dati sono considerati non validi, la pagina mostra un messaggio di errore sarà necessario reinserire i dati.

A registrazione completata si viene reindirizzati alla pagina di Login da cui reinserendo i propri dati si può accedere alla propria DashBoard

```
void "Registrazione"(){
    given: "Raggiungo la pagina di registrazione"
        go '/'
        $("a#registra").click()

    when: "L'utente immette credenziali non valide"
        $("input" , id:"username").value(" ")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "Si mostra un messaggio di errore sulla stessa pagina"
        $("p.inputError").text() == "Credenziali non valide"
        Utenti.findAll().size() == 0
        title == "Registrazione"

    when: "L'utente si registra con credenziali corrette"
        $("input" , id:"username").value("user")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "L'utente viene aggiunto al database e reindirizzato alla pagina di Login"
        Utenti.findAll().size() == 1
        title == "Login"
}
```

TEST DI INTEGRAZIONE

Accesso al Sistema

Accesso al sistema

Test di accettazione: AccessoAlSistemaSpec
Priorità: -
Punti Storia: -

Un utente senza account vuole registrarsi al sistema per accedere alle funzionalità della KanbanBoard.

Sarà necessario premere su un bottone dedicato alla registrazione e inserire le credenziali che preferisce.

Se i dati sono considerati non validi, la pagina mostra un messaggio di errore sarà necessario reinserire i dati.

A registrazione completata si viene reindirizzati alla pagina di Login da cui reinserendo i propri dati si può accedere alla propria DashBoard

```
void "Registrazione"(){
    given: "Raggiungo la pagina di registrazione"
        go '/'
        $("a#registra").click()

    when: "L'utente immette credenziali non valide"
        $("input" , id:"username").value(" ")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "Si mostra un messaggio di errore sulla stessa pagina"
        $("p.inputError").text() == "Credenziali non valide"
        Utenti.findAll().size() == 0
        title == "Registrazione"

    when: "L'utente si registra con credenziali corrette"
        $("input" , id:"username").value("user")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "L'utente viene aggiunto al database e reindirizzato alla pagina di Login"
        Utenti.findAll().size() == 1
        title == "Login"
}
```

TEST DI INTEGRAZIONE

Accesso al Sistema

Accesso al sistema

Test di accettazione: AccessoAlSistemaSpec
Priorità: -
Punti Storia: -

Un utente senza account vuole registrarsi al sistema per accedere alle funzionalità della KanbanBoard.

Sarà necessario premere su un bottone dedicato alla registrazione e inserire le credenziali che preferisce.

Se i dati sono considerati non validi, la pagina mostra un messaggio di errore sarà necessario reinserire i dati.

A registrazione completata si viene reindirizzati alla pagina di Login da cui reinserendo i propri dati si può accedere alla propria DashBoard

```
void "Registrazione"(){
    given: "Raggiungo la pagina di registrazione"
        go '/'
        $("a#registra").click()

    when: "L'utente immette credenziali non valide"
        $("input" , id:"username").value(" ")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "Si mostra un messaggio di errore sulla stessa pagina"
        $("p.inputError").text() == "Credenziali non valide"
        Utenti.findAll().size() == 0
        title == "Registrazione"

    when: "L'utente si registra con credenziali corrette"
        $("input" , id:"username").value("user")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "L'utente viene aggiunto al database e reindirizzato alla pagina di Login"
        Utenti.findAll().size() == 1
        title == "Login"
}
```

TEST DI INTEGRAZIONE

Accesso al Sistema

Accesso al sistema

Test di accettazione: AccessoAlSistemaSpec
Priorità: -
Punti Storia: -

Un utente senza account vuole registrarsi al sistema per accedere alle funzionalità della KanbanBoard.

Sarà necessario premere su un bottone dedicato alla registrazione e inserire le credenziali che preferisce.

Se i dati sono considerati non validi, la pagina mostra un messaggio di errore sarà necessario reinserire i dati.

A registrazione completata si viene reindirizzati alla pagina di Login da cui reinserendo i propri dati si può accedere alla propria DashBoard

```
void "Registrazione"(){
    given: "Raggiungo la pagina di registrazione"
        go '/'
        $("a#registra").click()

    when: "L'utente immette credenziali non valide"
        $("input" , id:"username").value(" ")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "Si mostra un messaggio di errore sulla stessa pagina"
        $("p.inputError").text() == "Credenziali non valide"
        Utenti.findAll().size() == 0
        title == "Registrazione"

    when: "L'utente si registra con credenziali corrette"
        $("input" , id:"username").value("user")
        $("input" , id:"password").value("1234")
        $("input" , id:"utenteCreate").click()

    then: "L'utente viene aggiunto al database e reindirizzato alla pagina di Login"
        Utenti.findAll().size() == 1
        title == "Login"
}
```

TEST DI INTEGRAZIONE

Creazione Progetto

Creazione Progetto

Test di accettazione: CreazioneProgettoSpec
Priorità: -
Punti Storia: -

Un utente registrato vuole creare una Board e le colonne di cui ha bisogno per gestire un progetto con due suoi colleghi: Davide e Giacomo, entrambi registrati nel sistema.

Se i nomi assegnati alla board e alle colonne sono validi, verrà mostrato una pagina con la propria board popolata dalle colonne, altrimenti i form di creazione mostreranno un messaggio di errore.

Una volta che l'utente ha creato la Board, la condivide con i suoi colleghi.

```
void "Creazione Board"(){
    when: "Premo su 'Nuova Board' e inserisco un nome non valido"
        $("a.create").click()
        $("input#nomeBoard").value(" ")
        $("input#createBoard").click()
    then: "Viene mostrato un messaggio di errore"
        $("p.inputError").text() != null
        Board.findByName(" ") == null
        title == "Crea nuova Board"

    when: "Premo su 'Nuova Board' e inserisco un nome valido"
        $("a.create").click()
        $("input#nomeBoard").value(nomeBoard)
        $("input#createBoard").click()
    then: "Viene mostrata la Board"
        Board.findAll().size() == 1
        title == "Board: " + Board.findByName(nomeBoard).name
}

void "Creazione colonne"(){
    given: "La board è stata creata"
        createBoard(nomeBoard)

    when: "L'utente crea una colonna con un nome non valido"
        createColumn(" ")
    then: "Viene mostrato un messaggio di errore"
        $("p.inputError").text() == "Il campo 'Nome della colonna' non può essere vuoto"
        Colonna.findAll().size() == 0
        title == "Crea colonna"

    when: "L'utente crea quattro colonne"
        createColumn("Backlog")
        createColumn("Ready")
        createColumn("Work in Progress")
        createColumn("Done")
    then: "Le colonne vengono mostrate nella home della Board"
        Colonna.findAll().size() == 4
        $("p.column-title" , 0).text() == "Backlog"
        $("p.column-title" , 1).text() == "Ready"
        $("p.column-title" , 2).text() == "Work in Progress"
        $("p.column-title" , 3).text() == "Done"
        title == "Board: " + nomeBoard
}
```

TEST DI INTEGRAZIONE

Creazione Progetto

Creazione Progetto

Test di accettazione: CreazioneProgettoSpec
Priorità: -
Punti Storia: -

Un utente registrato vuole creare una Board e le colonne di cui ha bisogno per gestire un progetto con due suoi colleghi: Davide e Giacomo, entrambi registrati nel sistema.

Se i nomi assegnati alla board e alle colonne sono validi, verrà mostrato una pagina con la propria board popolata dalle colonne, altrimenti i form di creazione mostreranno un messaggio di errore.

Una volta che l'utente ha creato la Board, la condivide con i suoi colleghi.

```
void "Aggiungi Collaboratori"(){
    given: "Esiste la Board e due collaboratori"
        createBoard(nomeBoard)

    when: "Aggiungo collaboratori da 'Gestisci'"
        $("a.adduser").click()
        $("form")."utenteSelezionato" = [username1 , username2]
        $("input" , name: "_action_aggiorna").click()
    then: "La board sarà condivisa tra il creatore e gli utenti aggiunti"
        title == "Board: " + nomeBoard
        Utenti.findAll().size() == 3
        Board.findByName(nomeBoard).users.size() == 3
}
```

TEST DI INTEGRAZIONE

Gestione Card

Gestione Card

Test di accettazione: GestioneCardSpec
Priorità: -
Punti Storia: -

Tre utenti che condividono la board “KanbanProject” hanno finito il task “Editing Colonna” e iniziato “Creazione Card”.

Per cui devono spostare le relative card alla colonna successiva e crearne una nuova su “Backlog” da assegnare a uno degli utenti

```
void "Sposta Card"(){
    given: "Esistono due Card nelle colonne 'Ready' e 'Work in Progress'"
        $("a p.column-addcard" , 1).click()
        createCard("Editing Colonna" , "Pagina per la modifica delle colonne")
        $("a p.column-addcard" , 2).click()
        createCard("Creazione Card" , "Form di creazione delle card")

    when: "Cambio colonna alle card dalla loro pagina di editing"
        $("p.card-titolo" , 1).click()
        $("form")."colonnaCard" = "Done"
        $("input" , name: "_action_update").click()
        $("p.card-titolo" , 0).click()
        $("form")."colonnaCard" = "Work in Progress"
        $("input" , name: "_action_update").click()
    then: "Le card sono spostate nella casella successiva"
        Card.findAll().size() == 2
        Colonna.findByName("Done").cards.size() == 1
        Colonna.findByName("Work in Progress").cards.size() == 1
        title == "Board: " + nomeBoard
}
```

```
void "Assegna Card"(){
    given: "Esiste una card su 'Backlog' da assegnare a un utente"
        def titoloCard = "Editing Card"
        $("a p.column-addcard" , 0).click()
        createCard(titoloCard , "Pagina per la modifica delle Card")

    when: "Cambio colonna alle card dalla loro pagina di editing"
        $("p.card-titolo" , 0).click()
        $("form")."utentiBoard" = ["Davide"]
        $("input" , name: "_action_update").click()
    then: "Le card sono spostate nella casella successiva"
        Card.findAll().size() == 1
        Colonna.findByName("Backlog").cards.size() == 1
        Card.findByTitolo(titoloCard).users.size() == 1
        Card.findByTitolo(titoloCard).users.getAt(0).username == "Davide"
        title == "Board: " + nomeBoard
}
```

TEST DI INTEGRAZIONE

Modifica e Cancellazione

Modifica e Cancellazione

Test di accettazione: ModificaECancellazione
Priorità: -
Punti Storia: -

Un utente deve riprendere a distanza di tempo un progetto, e decide di modificare la Board del progetto che non ha mai ultimato per poter riutilizzare lo schema delle colonne senza doverle ricreare.
Ha però bisogno di rinominare il nome della board in “KanbanProject” e il titolo errato di una colonna.

Dopo aver riadattato la board, l’utente decide di eliminare due colonne che non gli servono: “Testing”, “Ideas”.

Le card al loro interno verranno spostate in una colonna speciale chiamata “Card non assegnate”. L’utente eliminerà quelle che non gli servono e sposterà su “Backlog” quelle che vuole tenere

```
void "Rinomina Board"(){
    when: "Cambi il nome alla board"
    $("a.adduser").click()
    $("input#nome").value("KanbanProject")
    $("input.form-submit").click()

    then: "La board ha il nome aggiornato"
    title == "Board: KanbanProject"
}

void "Rinomina ed elimina colonne"() {

    when: "Rinomini la colonna Baclog in Backlog ed elimini Ideas
           (dalla edit) e Testing (dalla dashboard)"
        $("p.column-title", 1).click()
        $("input#nomeColonna").value("Backlog")
        $("input", name: "_action_update").click()

        $("p.column-title", 0).click()
        $("a.deleteButtonBar").click()

        $("input.deleteColumn", 3).click()

    then: "La Board contiene 5 colonne e una si chiama 'Backlog'"
        title == "Board: " + nomeBoard
        Colonna.findByName("Ideas") == null
        Colonna.findByName("Testing") == null
        Colonna.findByName("Backlog") != null
        Colonna.findAll().size() == 5
}
```

TEST DI INTEGRAZIONE

Modifica e Cancellazione

Modifica e Cancellazione

Test di accettazione: ModificaECancellazione
Priorità: -
Punti Storia: -

Un utente deve riprendere a distanza di tempo un progetto, e decide di modificare la Board del progetto che non ha mai ultimato per poter riutilizzare lo schema delle colonne senza doverle ricreare.
Ha però bisogno di rinominare il nome della board in “KanbanProject” e il titolo errato di una colonna.

Dopo aver riadattato la board, l’utente decide di eliminare due colonne che non gli servono: “Testing”, “Ideas”.

Le card al loro interno verranno spostate in una colonna speciale chiamata “Card non assegnate”. L’utente eliminerà quelle che non gli servono e sposterà su “Backlog” quelle che vuole tenere

```
void "Riassegna e cancella Card"(){
    given: "Il progetto non ha le colonne Ideas e Testing, ha Backlog"
        editBoardElements()

    when: "Elimina Card1 e Card2 e sposta Card3 da 'Card non Assegnate' a 'Backlog'"
        $("div.card-container" , 0).click()
        $("a.deleteButtonBar").click()

        $("input.deleteCard" , 0).click()

        $("div.card-container" , 0).click()
        $("form")."colonnaCard" = "Backlog"
        $("input", name: "_action_update").click()

    then: "La colonna Card non Assegnate non esiste più e Backlog ha una card"
        title == "Board: " + nomeBoard
        Colonna.findByName("Backlog").cards.size()==1
        Colonna.findByName("Card non assegnate") == null
        Colonna.findAll().size() == 4
        Card.findAll().size() == 1
        Card.findByTitolo("Card3") != null
}
```

TEST DI INTEGRAZIONE

Eliminazione Board e Logout

Eliminazione Board e Logout

Test di accettazione: EliminazioneBoardLogout
Priorità: -
Punti Storia: -

Un utente dopo aver passato molto tempo a sviluppare il suo prodotto, ha completato tutti i task necessari.

Decide quindi di eliminare la board per lasciare nella sua dashboard solo i progetti aperti, ed effettua poi il logout

```
void "Eliminazione Board"(){
    when: "Elimino la Board"
        $("a.adduser").click()
        $("a.deleteButtonBar").click()

    then: "Viene visualizzata la dashboard senza la Board eliminata"
        find("p.nomeBoard").isEmpty()
        title == "Dashboard"

}

void "Logout"(){
    when: "Premo sul bottone Logout"
        $("a#logout").click()

    then: "Vengo reindirizzato alla pagina di login"
        title == "Login"

}
```