UNIVERSITÀ DI TRENTO

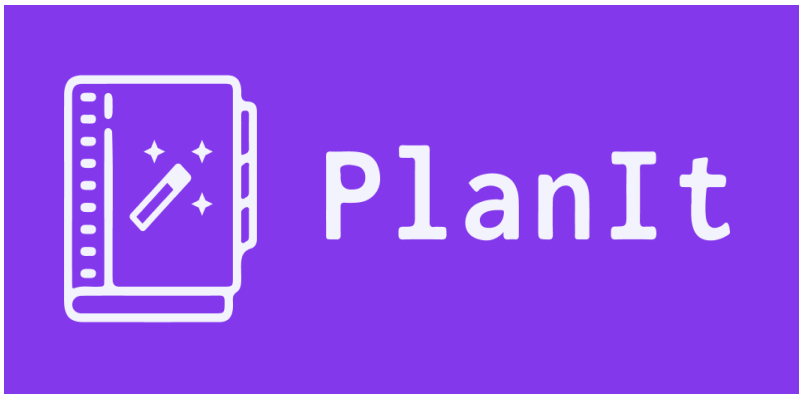
Dipartimento d'Ingegneria e
Scienze dell'informazione

Progetto:

PlanIt

Titolo del documento:

Documento di Architettura (Diagrami delle classi e codice in OCL)



Gruppo T56
Gabriele Lacchin, Denis Lucietto ed Emanuele Zini

Indice

Scopo del documento

Il presente documento riporta la definizione dell'architettura del progetto PlanIt usando diagrammi delle classi in Unified Modeling Language (UML) e codice in Object Constraint Language (OCL). Nel precedente documento D2 è stato presentato il diagramma degli use case, il diagramma di contesto e quello dei componenti con le relative descrizioni e specificazioni. Ora, tenendo conto di questa progettazione, viene definita l'architettura del sistema dettagliando da un lato le classi che dovranno essere implementate a livello di codice e dall'altro la logica che regola il comportamento del software. Le classi vengono rappresentate tramite un diagramma delle classi in linguaggio UML. La logica viene descritta in OCL perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML. Tutti i diagrammi, che verranno presentati, hanno una descrizione a loro associati per delinearne il loro scopo e funzionalità.

- diagramma delle classi;
- object constraint language;
- diagramma delle classi e codice object constraint language.

1 Diagramma delle classi

Nel presente capitolo vengono presentate le classi previste nell'ambito del progetto PlanIt. Ogni componente presente nel diagramma dei componenti è stata utilizzato per fare una o più classi. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi che identificano i dati gestiti dalla classe e una lista di metodi che definiscono le operazioni previste all'interno della classe. Ogni classe può essere anche associata ad altre classi e, tramite questa associazione, è possibile fornire informazioni su come le classi si relazionano tra loro. Anche se, sottolineiamo, che le associazioni fornite tra le classi sono quelle non banali e che non richiedono un'ulteriore specificazione. Riportiamo di seguito le classi individuate a partire dai diagrammi di contesto e dei componenti. In questo processo si è proceduto anche nel massimizzare la coesione e minimizzare l'accoppiamento tra classi, cercando di evitare la ridondanza.

1 : **Tipologie di utenti**
Analizzando il diagramma di contesto realizzato per il progetto PlanIt si nota la presenza di tre attori: "utente autenticato standard", "utente non autenticato" e "utente autenticato premium", per questo motivo sono state delineate tre classi, ovvero "Utente", "Utente Autenticato", "Utente Non Autenticato". Si è deciso di fare la classe "Utente" in modo tale di unire le due tipologie di utenti, "Utente Autenticato" e "Utente Non Autenticato", che si legano a "Utente" con delle generalizzazioni. L'unica funzionalità che ha "Utente" è il "Logout()" che non è altro che l'azione che riporta l' "Utente" nell' Homepage del PlanIt; le funzionalità, che possono essere fatte nella pagina Homepage, verranno descritte nella presentazione della classe "Homepage" (in DCL1.8, da mettere riferimento). L' "Utente Non Autenticato" è colui che utilizza il sito in versione demo; che, come abbiamo già descritto in precedenza nel D1 e D2, ha la possibilità di usare il sito con delle limitazioni. Queste funzionalità sono presenti in "Gestione Salvataggio Dati In Locale": l'importanza di questa classe è anche quella di salvare in locale le modifiche effettuate dall' "Utente Non Autenticato", infatti nel caso in cui questo decidesse di autenticarsi nel sito, le modifiche effettuate nella versione demo, verranno trasferite e salvate nel suo account di "Utente Autenticato". L' "Utente Autenticato" è colui che accede al sito autenticandosi, ottenendo tutte le funzionalità del sito, avendo delle limitazioni rispetto all' utente autenticato che si abbona al sito (D1-RF4.2). La gestione della sottoscrizione al sito e quindi il passaggio all'account utente autenticato premium è gestito da "Impostazioni Account", che verrà presentato successivamente, invece la gestione dell'autenticazione viene gestita dalla classe "Interfaccia Auth0" a cui è legata mediante un'associazione. Questa classe viene descritta nel paragrafo successivo. Gli unici attributi che ha "Utente Autenticato" sono i suoi dati più importanti, ovvero: "UserID", "Email" e "Username". Questi attributi, ottenuti da Interfaccia di Auth0 dopo l'autenticazione, insieme ad altre impostazioni dell'account, sono salvate nel database MongoDB. Per questo motivo, "Utenti Autenticato" è legato con "Gestione Chiamate MongoDB" che verrà descritto in seguito.

2 : **Interfaccia Auth0**
Il diagramma di componenti presentato nel documento D2 presenta un componente chiamato allo stesso modo. Questa classe rappresenta il meccanismo di autenticazione degli utenti attraverso un sistema esterno presente anche nel diagramma di contesto, ovvero Auth0. Questa classe si interfaccia con questo sistema esterno di autenticazione, che gestirà completamente la procedura di autenticazione. Il software di PlanIt memorizzerà solo l'email, UserId e Username, valori restituiti dall'autenticazione che verranno salvati anche nella classe, presentata precedentemente, "Utente Autenticato", istanziandola. Questa classe fa partire la procedura di autenticazione per un "Utente" una volta invocata dalla funzione "AccessoWebApp()" presente nella classe "Homepage", descritta in DCL1.8, o dalla funzione "Autenticazione()" presente nella classe "Utente Non Autenticato". Ricordiamo, che per autenticazione intendiamo o registrazione o login, a seconda se l' "Utente" abbia o meno un suo account su PlanIt.

3 : **Impostazioni Utente/ Scherma "Impostazioni Account"**
Il diagramma di componenti analizzato presenta un componente "Gestione impostazioni account" e altri componenti utilizzati per gestire le impostazioni dell'account, dunque è stato identificato la classe "Impostazioni Utente" che permette all'utente autenticato, insieme alle classi a cui è associato, di modificare e gestire il proprio account secondo le proprie preferenze per le seguenti impostazioni, nella pagina "Impostazioni Account": sistema di mappe

(Impostazioni utente?)

preferito da utilizzare nell'aggiunta del luogo dell'evento, ore di sonno, sottoscrizione all'abbonamento (divenendo "utente autenticato premium"), scelta metodo di pagamento, scelta metodo di utilizzo Google Calendar, modifica password ed username. "Impostazioni Utente" è collegato a "Utente Autenticato" in modo tale che ogni volta che l'utente autenticato va a modificare delle proprie impostazioni account, queste sono salvate su MongoDB; infatti, ricordiamo, che "Utente Autenticato" è collegato alla classe "Gestione chiamate MongoDB" che verrà descritto successivamente.

3.1 : **Gestione modifica password e Auth0 management API**
Analizzando il diagramma di componenti si nota la presenza di "Gestione modifica password" e "Interfaccia management API Auth0", per questo motivo sono state fatte queste classi con gli stessi nomi, il cui loro scopo è gestire il modifica password, azione che può essere fatta dalle impostazioni account, e il reset password, azione che si può fare nel caso in cui si vuole recuperare e cambiare la password nel caso in cui si è dimenticata partendo dalla pagina di autenticazione di PlanIt. La nuova password inserita viene controllata in "Gestione modifica password", in modo tale che la password segui le regole delineate in D1-RNF2. Una volta che la nuova password passa il controllo, viene inviata ad Auth0 grazie alla funzione "CambiaPassword(UserId, nuovaPassword) presente in "Auth0 management API".

3.2 : **Abbonamento**
Il diagramma di contesto analizzato nel documento D2 presenta un terzo attore che è stato già citato in DCL1.1, ovvero l'utente autenticato premium, che usufruisce di un abbonamento al sito accedendo alla versione premium della piattaforma, per questo motivo è stata fatta la classe "Abbonamento" il cui scopo è quello di gestire la sottoscrizione e cancellazione dell'abbonamento. Inoltre, grazie a questa classe, è possibile sapere se un utente autenticato sia standard (attributo Abbonato = false) o premium (Abbonato = true). Come si può notare è presente una composition tra "Impostazioni Utente" e "Abbonamento" il cui scopo è indicare che la classe "contenuta" "Abbonamento" esiste solo con la classe "contenitrice" "Impostazioni Utente"; infatti la classe "Abbonamento" ha senso ed esiste solo se l'utente ha le "Impostazioni Utente", ovvero che sia autenticato.

3.3 : **Gestione Google Calendar**
Analizzando il diagramma di componenti realizzato per il progetto PlanIt si nota la presenza dei componenti "Gestione utilizzo Google Calendar", "Gestione sincronizzazione" e "Gestione import/export file", per questo motivo sono state fatte queste tre classi il cui scopo è quello di gestire l'integrazione ed interazione di PlanIt con Google Calendar. "Gestione Google Calendar" contiene l'attributo "sincronizzazione" il cui scopo è quello di salvare quale sia il metodo di integrazione di Google Calendar utilizzato dall'utente autenticato. Inoltre, questa classe permette il cambio della tipologia di interazione con Google Calendar usufruendo delle classi "Gestione Import/Export" e "Gestione Sincronizzazione" che gestiscono i processi di import/export di file di eventi o calendar di Google Calendar da e/o verso PlanIt e la sincronizzazione (l'account Google viene collegato con quello PlanIt) con Google Calendar, con cui avviene l'importazione ed esportazione di eventi e calendari automaticamente, rispettivamente.

3.4 : **Gestione sistema di pagamento**
Il diagramma dei componenti analizzato, realizzato per il progetto PlanIt, presenta i componenti "Gestione metodo di pagamento", "Interfaccia PayPal" e "Interfaccia Payments", per questo motivo sono state pensate queste cinque classi: "Sistema Di Pagamento", "Sistema di Pagamento Stripe", "Sistema di Pagamento PayPal", "PayPal API" e "Stripe API". La funzione di queste classi è gestire il pagamento dell'abbonamento e cambiamento del metodo di pagamento. "Sistema Di Pagamento" salva il giorno di fatturazione dell'abbonamento da parte dell'utente autenticato premium e permette, come si nota, il cambiamento del metodo di pagamento, il quale può esser scelto tra PayPal e Stripe. "Sistema di Pagamento PayPal", "Sistema Di Pagamento Stripe, "PayPal API" e "Stripe API" gestiscono il processo di pagamento dell'abbonamento mediante il metodo di pagamento "PayPal" e "Stripe" rispettivamente conservando, una volta inserite, le credenziali di pagamento per questi sistemi dell'utente autenticato premium.

4 : Gestione schermata "Calendario"/ Visualizzazione Calendari

Analizzando il diagramma di componenti fatto per il progetto PlanIt si può vedere la presenza di "Gestione visualizzazione calendari ed eventi", "Gestione Attività", "Gestione filtro impegni" e "Gestione filtri calendari". Per questa ragione sono state ideate le seguenti classi: "Visualizzazione Calendari", "Visualizzazione Informazioni Evento", "Gestione filtro Eventi", "Gestione Filtri Calendari" e "Gestione Attività". Queste classi permettono la visualizzazione di calendari ed eventi nella schermata Calendario (D1-FE2) anche visualizzandoli seguendo dei filtri che può inserire l'utente. "Visualizzazione Calendari" ottiene da "Gestione chiamate MongoDB" i calendari dell'utente che dopo mostra a quest'ultimo. Dopo, grazie a "Gestione Filtri Calendari" vengono mostrati i calendari secondo dei filtri. In seguito, "Gestione filtro Eventi" mostra una lista di eventi secondo dei criteri delineati dall'utente. Infine, "Gestione Attività" permette all'utente di visualizzare la lista degli eventi giornalieri, permettendo a quest'ultimo di segnare il completamento o meno dell'attività o anche l'eliminazione di questa. L'eliminazione o l'indicazione di non completamento dell'attività porta alla rischedulazione del calendario.

5 : Dashboard

Il diagramma dei componenti esibisce i seguenti componenti: "Gestione Dashboard", "Gestione eventi settimanali" e "Gestione eventi giornalieri". Dunque, sono state progettate le classi "Dashboard", "Dashboard Eventi Settimanali" e "Dashboard Eventi Giornalieri", il cui fine è quello di mostrare dei grafici per fornire delle informazioni sull'uso del tempo all'utente. La classe "Dashboard" ottiene da "Gestione Chiamate MongoDB" i calendari di un utente, in modo tale da poter avere anche tutti i suoi eventi. Grazie agli eventi mostra una HeatMap, il cui scopo è già stato descritto in D2-US8, e passa gli eventi settimanali a "Dashboard Eventi Settimanali" e passa gli eventi giornalieri a "Dashboard Eventi Giornalieri" in modo tale che questi possano mostrare altri grafici riguardo gli eventi nel periodo di tempo considerato mediante i grafici citati in D2-US8 (grafico a barre per "Dashboard Eventi Settimanali" e grafico a torta con lista di eventi per "Dashboard Eventi Giornalieri").

6 : Gestione evento e calendario

Il diagramma dei componenti analizzato, realizzato per il progetto PlanIt, presenta i componenti "Gestione creazione o modifica evento" e "Gestione creazione e modifica calendario" e dei componenti a loro associati. Per questo motivo, sono state fatte le classi "Calendario" e "Evento" e le classi a loro associate, che si possono vedere nella foto sottostante, che hanno l'obiettivo di permettere all'utente di poter creare o modificare eventi e calendari.

6.1 : Gestione creazione o modifica calendario

Analizzando il diagramma di componenti fatto per il progetto PlanIt si può vedere la presenza della componente "Gestione creazione o modifica calendario", per questa ragione sono state ideate le classi "Impostazioni Predefinite Eventi" e "Calendario" che gestiscono la procedura di creazione o modifica del calendario secondo le impostazioni scelte dall'utente; quando l'utente va a modificare o creare un calendario può andare a definire anche dei valori con cui andare a precompilare alcuni campi presenti nella creazione o modifica evento, lo scopo di "Impostazioni Predefinite Eventi" è quello di gestire la procedura di definizione di questi campi, andando anche a salvare i loro valori. Una volta che un calendario è stato creato o modificato, questo viene salvato su MongoDB grazie alla classe "Gestione chiamate MongoDB" che lo invia al sistema esterno di archiviazione MongoDB (si noti l'associazione tra "Calendario" e "Gestione chiamate MongoDB"). "Gestione chiamate MongoDB" viene presentato più nel dettaglio successivamente. Si specifica, che i vari set e get degli attributi presenti in "Impostazioni Predefinite Eventi" e "Calendario" non sono stati scritti nelle funzioni di tali classi in quanto ritenute banali.

6.2 : Gestione creazione o modifica evento

(ricordati di menzionare impostazioni predefinite eventi) → ottenute dal calendario
Analizzando il diagramma di componenti, fatto per il progetto PlanIt, si può vedere la presenza dei componenti "Gestione creazione o modifica evento" e "Gestione notifiche",

per questa ragione sono state ideate le classi “Notifica” e “Evento”. La seconda classe citata gestisce la procedura di creazione o modifica dell’evento secondo dei valori scelti per i vari attributi presenti nella classe “Evento”. Invece, la classe “Notifica” gestisce la procedura di creazione o modifica della notifica per quel specifico evento che si sta creando o modificando, per questo motivo è presente un’associazione tra “Evento” e “Notifica”. Come si può notare è presente una “composition” tra “Evento” e “Calendario” che serve ad indicare che un evento, “oggetto contenuto”, esiste solo con l’oggetto “contenitore” “Calendario”, infatti un evento non può non essere non associato ad un calendario. Sono state definite anche le classi “Evento Singolo”. “Evento Ripetuto”, “Impostazioni Ripetizione Base”, “Impostazioni Ripetizione Avanzate” e “Giornata” che hanno la funzionalità di dirigere il processo di definizione temporale di quando avviene l’evento che si sta modificando o creando. Sono state individuate due classi distinte “Impostazioni Ripetizione Base” e “Impostazioni Ripetizione Avanzate” per andare a gestire la definizione di un evento ripetuto, poiché la prima classe citata viene utilizzata per sovrintendere la determinazione di eventi di cui viene definito un numero di volte in cui viene ripetuto senza indicare quali giornate, e la seconda classe, al contrario, viene utilizzata per sovrintendere la determinazione di eventi ripetuti dei quali si indicano anche le giornate. Si specifica, che i vari set e get degli attributi presenti in “Evento”, “Notifica”, “Evento Ripetuto”, Evento Singolo”, “Impostazioni Ripetizione Base”, “Impostazioni Ripetizione Avanzate” e “Giornata” non sono stati scritti nelle funzioni di tali classi in quanto ritenute banali.

6.3 : Gestione Luoghi

Il diagramma dei componenti analizzato, realizzato per il progetto PlanIt, presenta un componente “Interfaccia mappe”, per questa ragione sono state fatte le classi “Gestione Luoghi”, “Google Maps API” e “OpenStreetMap API”, che hanno lo scopo di gestire la definizione delle coordinate del luogo dove avviene un evento. Poiché la definizione del luogo di eventi può esser definito sia dalle “Impostazioni predefinite Eventi” in fase di creazione o modifica calendario (il luogo appartiene ad un dei campi che può esser precompilato) e può esser definito anche in fase di creazione o modifica evento, questa classe “Gestione Luoghi” è legata sia a “Impostazioni Predefinite Eventi” sia a “Evento”. Sono state inserite due classi “Google Maps API” e “OpenStreetMap API”, che hanno lo scopo di gestire la procedura di conversione dell’indirizzo del luogo inserito in coordinate mediante l’utilizzo delle piattaforme “Google Maps” e “OpenStreetMap” rispettivamente. Infatti, si ricorda, che l’utente autenticato può decidere, secondo le proprie preferenze, da impostazioni account (DCL1.3) quale sistema di mappe utilizzare.

6.4 : Gestione condivisione

Analizzando il diagramma di componenti, fatto per il progetto PlanIt, si può vedere la presenza della componente “Gestione condivisione”, allora è stata fatta la classe “Gestione Condivisione", che ha la funzionalità di gestire la procedura di condivisione di calendari ed eventi; infatti, come si può osservare, questa classe ha un’ “association” sia con “Evento” che con “Calendario”. La condivisione viene effettuata utilizzando l’email dei partecipanti a cui condividere l’evento o il calendario e un link contenente le informazioni fondamentali riguardanti l’ “Evento” o “Calendario” che si sta condividendo. Nel link che viene inviato, è presente: IDrichiesta, IDEvento o IDCalendario. L’IDrichiesta è fondamentale per poter individuare univocamente la richiesta che è stata inviata, invece gli IDEvento o IDCalendario sono fondamentali per definire cosa si sta condividendo. Infatti, nel caso in cui si accettasse la richiesta, il sistema deve sapere a quale oggetto “Evento” o “Calendario”, presente su MongoDB (in quanto questi oggetti vengono salvati su questo sistema esterno di archiviazione), si sta facendo riferimento; infatti, nel database c’è una singola istanza per ciascuno oggetto che è stato condiviso. Dunque, grazie all’email del partecipante e tale link che è stato descritto, questa classe invia la richiesta di condivisione, salvando l’esito di tale richiesta nell’attributo booleano “Accettata”. Nel caso in cui la richiesta fosse accettata, questi “Utenti Autenticati”, che hanno accettato la condivisione, vengono aggiunti nell’attributo "Partecipanti" di o “Evento” o “Calendario” a seconda di cosa si sta condividendo.

Email invito → link parametri di tutto, a che calendario, id evento o id richiesta, o calendario o evento. Quando accetti vieni inserito nella lista. Id calendario o evento e

ti aggiunge. Singola istanza per calendario o evento condiviso.

7 : **Interfaccia Cloudflare**

Analizzando il diagramma di contesto realizzato per il progetto PlanIt si nota la presenza del sistema esterno "Cloudflare", per questa ragione è stata fatta questa classe "Interfaccia CloudFlare" che permette ottenere il log di tutte le connessioni che sono state effettuate per accedere al sito con il resoconto se sono state filtrare o meno. Per ottenere tali informazioni si interfaccia al sistema esterno Cloudflare. Classe.

8 : **Gestione Homepage**
Il diagramma dei componenti analizzato, realizzato per il progetto PlanIt, presenta le componenti "Gestione Homepage" e "Interfaccia Iubenda", dunque sono state ideate le classi "Interfaccia Iubenda", "Gestione Homepage" e "Tipo_Cookie". La classe "Gestione Homepage" gestisce la procedura di scelta, effettuata da un utente che vuole entrare nel sito, tra entrare nel sito in modalità demo, accedendo da utente non autenticato, o entrare nella web app autenticandosi, ovvero registrandosi o accedendo tramite Auth0 su PlanIt. Infatti, qualora l'utente abbia deciso di accedere alla web app con l' autenticazione verrà indirizzato alla pagina di gestione login e registrazione di Auth0 grazie all' "Interfaccia di Auth0", attraverso la funzione "RedirectToAuthentication()"; ad autenticazione avvenuta verrà indirizzato alla web app. Dunque, questa classe gestisce l'indirizzamento dell'utente nella modalità del sito scelta. Inoltre, "Gestione Homepage" mostra anche il banner di cookie contenente le cookie policy che possono essere accettate o meno dall'utente. Il banner di cookie, come anche le politiche di privacy, sono gestite e ottenute dalla classe "Interfaccia Iubenda", a cui "Gestione Homepage" è legata. "Interfaccia Iubenda", interfacciandosi con Iubenda, invia all' "Homepage" il codice del banner di cookie contenente anche il link alla pagina di politiche di privacy. Una volta accettati i cookie, questi vengono salvati nell'Homepage nell'attributo "Cookie", il cui tipo è "Tipo_cookie", che rappresenta i cookie salvati nel browser dell' Utente.

(metto tutto assieme senza fare una descrizione a parte per Iubenda e Tipo_Cookie)

9 : **Tipo calendario filtrato**

Nella classe "Gestione Filtri Calendari" è presente un attributo di tipo "Tipo_Calendario_Filtrato", tipo di dato che non avevamo ancora descritto. La classe "Tipo_Calendario_Filtrato" è stata delineata per definire il tipo di calendario che viene utilizzato nella classe "Gestione Filtri Calendari". Questo tipo permette di andare a selezionare e deselezionare un calendario, azioni fondamentali per andare a filtrare i calendari.

10 : **Tipo Fuso orario**

Nella classe "Calendario" è presente un attributo di tipo "Tipo_Fuso_Orario", tipo di dato che non avevamo ancora descritto. La classe "Tipo_Fuso_Orario" è stata delineata per definire il tipo di dato dell'attributo "FusoOrario": questo tipo permette di salvare, prendere (get) e impostare (set) l'offset del fuso orario del calendario che stiamo andando a creare o modificare e iil nome della località che ha tale fuso orario. L'intero presente in "GMToffset" è un offset rispetto al GMT, ovvero l'orario di Greenwich, dunque l'attributo salvato è ad esempio: "+1", "+2", "-1", e così via. Le funzione set() e get() di "GMToffset" e "Località" non sono state inserite nel diagramma della classe perché ritenute banali. Per vedere l'utilità della funzionalità del fuso orario di un calendario, si legga D2-US10.

11 : **Tipo_Luogo**

Nelle classi "Impostazioni Predefinite Eventi" e "Evento" è presente l'attributo "Luogo" di tipo "Tipo_Luogo", tipo di dato che non avevamo ancora descritto. Questo tipo permette di salvare il valore della "Latitudine" e "Longitudine", ovvero le coordinate, del luogo dove avviene un evento. La definizione del luogo, dove avviene un evento, è presente nella classe creazione o modifica di un evento o nelle impostazioni predefinite di calendario, dove andiamo a definire il luogo con cui andare a precompilare gli eventi che stiamo creando appartenenti

a quel calendario. Il valore della latitudine e longitudine di un luogo viene restituito dalla funzione "ConvertiIndirizzoToCoordinate(Indirizzo)" presente in "Gestione Luoghi", classe già presentata.

12 : **Tipo Data**
Nella classe "Notifica" è presente l'attributo "Data" di tipo "Tipo_Data", tipo di dato che non avevamo ancora descritto. Questo tipo di dato permette di andare a salvare, definire e ottenere il valore di ora, minuti, giorno, mese anno di quando deve essere inviata una notifica. Le funzione set() e get() di "Ora", "Minuti", "Giorno", "Mese" e "Anno" non sono state inserite nel diagramma della classe perché ritenute banali.

13 : **Gestione chiamate MongoDB**
Il diagramma dei componenti analizzato, realizzato per il progetto PlanIt, presenta un componente "Gestione chiamate MongoDB", per questo motivo è stata fatta questa classe il cui scopo è gestire, come dice il nome, le chiamate al sistema esterno MongoDB. Queste interazioni con MongoDB, possono essere sia di accesso a contenuti presenti nel database MongoDB, sia invio di dati a quest'ultimo. I dati che vengono salvati su MongoDB, e quindi gestiti nella chiamate di questa classe, riguardano: i calendari, eventi e l'account (ovvero, userID e email, username e le impostazioni utente, ovvero gli attributi presenti in "Gestione Impostazioni Utente" di un utente autenticato), per questa ragione sono presenti, in questa classe, le relative funzioni per permettere l'invio e la richiesta di ricezione di questa tipologia di dati.

2 Object Constraint Language

1 : Utenti

```
CONTEXT Utente_Non_Autenticato::Autenticazione() :
POST : UserId <> "" AND Email <> "" AND Username <> ""
```

```
CONTEXT Utente_Autenticato :
INV : UserId <> "" AND Email <> "" AND Username <> ""
```

```
CONTEXT Utente::Logout() :
PRE :  UserId <> "" AND Email <> "" AND Username <> ""
--CONTEXT Utente::Logout() :
POST : UserId  = "" AND Email  = "" AND Username  = ""
```

da chiedere

2 : Auth0

```
CONTEXT Interfaccia_Auth0::RedirectToAutenticationPage() :
POST : Utente_Autenticato.UserId = AUserId AND
    Utente_Autenticato.Email = AEmail AND Utente_Autenticato.Username
    = AUsername
```

3 : Impostazioni Utente

```
CONTEXT ImpostazioniUtente:
INV : (SistemaMappe = "OSM" OR SistemaMappe = "GM") AND OreSonno >= 0
-- AND Utente_Autenticato.UserId <> "" AND Utente_Autenticato.Email
    <> "" AND Utente_Autenticato.Username <> ""
-- AND Premium = Abbonamento.Abbonato

CONTEXT ImpostazioniUtente::CambiaUsername(NUsername):
PRE : NUsername <> ""
POST : Utente_Autenticato.Username = NUsername

CONTEXT ImpostazioniUtente::CambiaNumeroOreSonno(Nore):
PRE : NOre >= 0 AND NOre <> self.OreSonno
POST : self.OreSonno = NOre

--CONTEXT ImpostazioniUtente::EliminaAccount():

CONTEXT ImpostazioniUtente::GetSistemaMappe():
POST : return = SistemaMappe
-- Da chiedere

CONTEXT ImpostazioniUtente::CambiaSistemaMappe(NSistemaMappe):
PRE : (NSistemaMappe = "OSM" AND self.SistemaMappe = "GM") OR
    (NSistemaMappe = "GM" AND self.SistemaMappe = "OSM")
POST : self.SistemaMappe = NSistemaMappe
```

da chiedere

4 : Gestione Modifica Password

```
CONTEXT GestioneModificaPassword :
INV : NuovaPassword <> ""
```

```
CONTEXT GestioneModificaPassword::ControlloValidiaPassword(NPassword)
    :
PRE : NPassword.exists(carattere | carattere = [a, ..., z]) AND
    NPassword.exists(carattere | carattere = [A, ..., Z]) AND
    NPassword.exists(carattere | carattere = [0, ..., 9]) AND
    NPassword.exists(carattere | carattere = [!, #, $, %, &, *])

-- CONTEXT GestioneModificaPassword::SetPassword(NPassword) :
-- PRE : NPassword <> ""
-- POST : self.NuovaPassword = NPassword


CONTEXT GestioneModificaPassword::ControlloSetPassword(NPassword) :
PRE : NuovaPassword <> "" AND self.ControlloValidiaPassword(NPassword)
POST : self.NuovaPassword = NPassword
```

5 : Auth0 management API

```
CONTEXT AuthOManagementAPI :
INV : EndPoint <> "" AND AuthorizationToken <> ""

CONTEXT AuthOManagementAPI::CambiaPassword(UserId, NuovaPassword) :
PRE : UserId <> "" AND
    GestioneModificaPassword.ControlloValidiaPassword(NuovaPassword)
```

6 : Abbonamento

```
CONTEXT Abbonamento::SottoscrizioneAbbonamento() :
PRE : Abbonato = false
POST : Abbonato = true

CONTEXT Abbonamento::CancellazioneAbbonamento() :
PRE : Abbonato = true
POST : Abbonato = false
```

7 : GoogleCalendar

```
CONTEXT Abbonamento::CambiaTipologiaInterazione() :
POST : sincronizzazione = NOT sincronizzazione@pre
```

8 : Visualizzazione calendari ed eventi

```
CONTEXT VisualizzazioneCalendariEventi :
INV

CONTEXT
    VisualizzazioneCalendariEventi::CambiaIntervalloVistaUtenti(NIntervallo)
    :
POST : IntervalloVistaEventi = NIntervallo
```

9 : Privacy

```
CONTEXT VisualizzazioneCalendariEventi :
INV
```

10 : Calendario

```
CONTEXT Calendario :
INV : IdCalendario <> "" AND Nome <> "" AND Proprietario <> Null

CONTEXT Calendario::Crea(Principale)
POST : Proprietario = UserId

CONTEXT Calendario::Salva(NNome, NFuso, NColore)
POST : Nome = NNome AND Fuso = NFuso AND NColore = Colore
```

11 : Impostazioni Predefinite Eventi

12 : Gestione Condivisione

13 : Evento

```
CONTEXT Evento :
INV : IdEvento <> "" AND Nome <> "" AND Proprietario <> Null AND
    Titolo <> ""

CONTEXT Evento::Crea(NTitolo)
PRE : Ntitolo <> ""
POST : Proprietario = UserId AND Titolo = NTitolo

CONTEXT Evento::Salva(NTitolo, NDescrizione, NPriorita, NDifficolta,
    NLuogo)
PRE : Ntitolo <> ""
POST : Titolo = NTitolo AND Descrizione = NDescrizione AND Priorita =
    NPriorita AND Difficolta = NDifficolta AND Luogo = NLuogo AND
```

3 Diagramma delle classi e codice Object Constraint Language