ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Management by Objectives

Elaborato in
Applicazioni e servizi web

Autori
Davide Alpi
Angelo Parrinello
Paolo Penazzi

Anno Accademico 2022-2023

Introduzione

Il progetto consiste nella creazione di un software di Management By Objectives (MBO).

In sostanza, si tratta di un software che permette di definire obiettivi per i dipendenti di un'azienda, e di monitorarne il raggiungimento.

In base al raggiungimento degli obiettivi, ai dipendenti spetta un premio in denaro.

Requisiti

L'azienda utilizza attualmente un semplice foglio excel, con una vista alquanto confusionaria, e senza la possibilità per ciascun dipendente di vedere solo i propri dati.

Il software deve permettere di definire obiettivi per l'anno corrente per i dipendenti di un'azienda, e di monitorarne il raggiungimento.

Un obiettivo è definito in questo modo:

- Nome, univoco e significativo
- Tipologia, gli obiettivi possono essere "Aziendale" (comune a tutti, come il fatturato), "Funzione" (relativo allo specifico ruolo del dipendente), o "Complementare" (un obiettivo raggiungibile dal ruolo del dipendente in collaborazione con altri dipartimenti aziendali), o "Soft Skills" (obiettivi di crescita personale).
- Base, il valore di partenza dell'obiettivo.
- 100%, il valore che l'obiettivo deve raggiungere entro fine anno per ottenere il minimo dei punti relativi a quest'obiettivo.
- 150%, il valore che l'obiettivo deve raggiungere entro fine anno per ottenere il massimo dei punti relativi a quest'obiettivo.
- Peso, il peso dell'obiettivo rispetto agli altri obiettivi. Gli gli obiettivi assegnati ad un dipendente deve sommare a 100.
- Uptodate, il valore attuale dell'obiettivo (i dati possono venire aggiornati man mano durante l'anno).

I dipendenti sono modellati in maniera molto semplice, con username, password, nome, cognome e ruolo aziendale. L'applicativo deve permettere al manager (super-user) di:

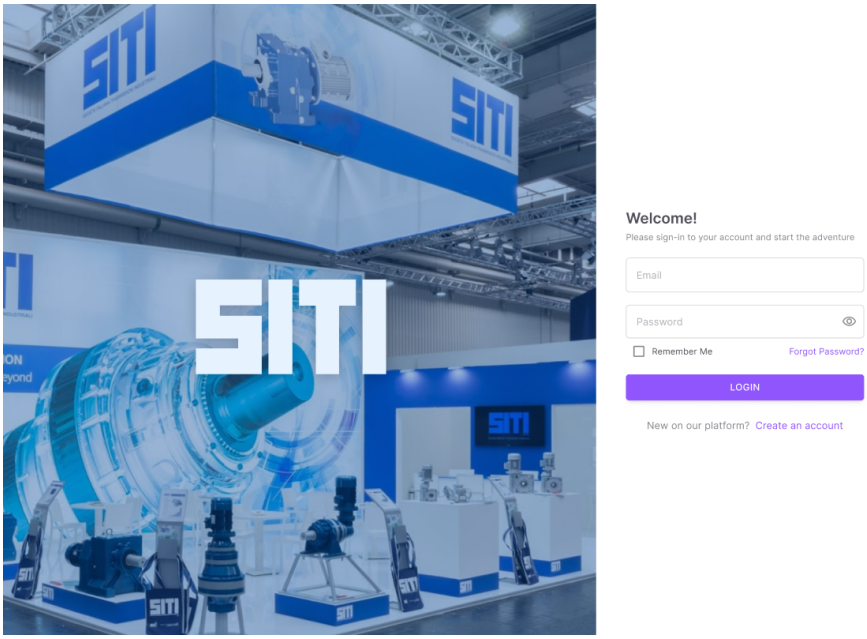

Figure 1: Schermata di login

- Creare l'anagrafica dei dipendenti.
- Caricare gli obiettivi per l'anno corrente, assegnati a ciascun dipendente.
- Caricare i valori up to date, sulla base dei quali verrà calcolato il punteggio.

L'applicativo deve permettere al dipendente di:

- Vedere i propri obiettivi, e il loro stato di avanzamento.
- Simulare come cambierebbe il punteggio se l'obiettivo raggiungesse un certo valore.

Entrambe le tipologie di utenti chiaramente devono poter loggarsi nell'applicativo. Per il momento il software non deve permettere la registrazioni di nuovi utenti, che verranno configurati manualmente.

Design

Mockup

Il valore di chiusura è il valore raggiunto per l'obiettivo a fine anno. Di default è pari al valore base per l'anno corrente, finchè i dati up-to-date non vengono

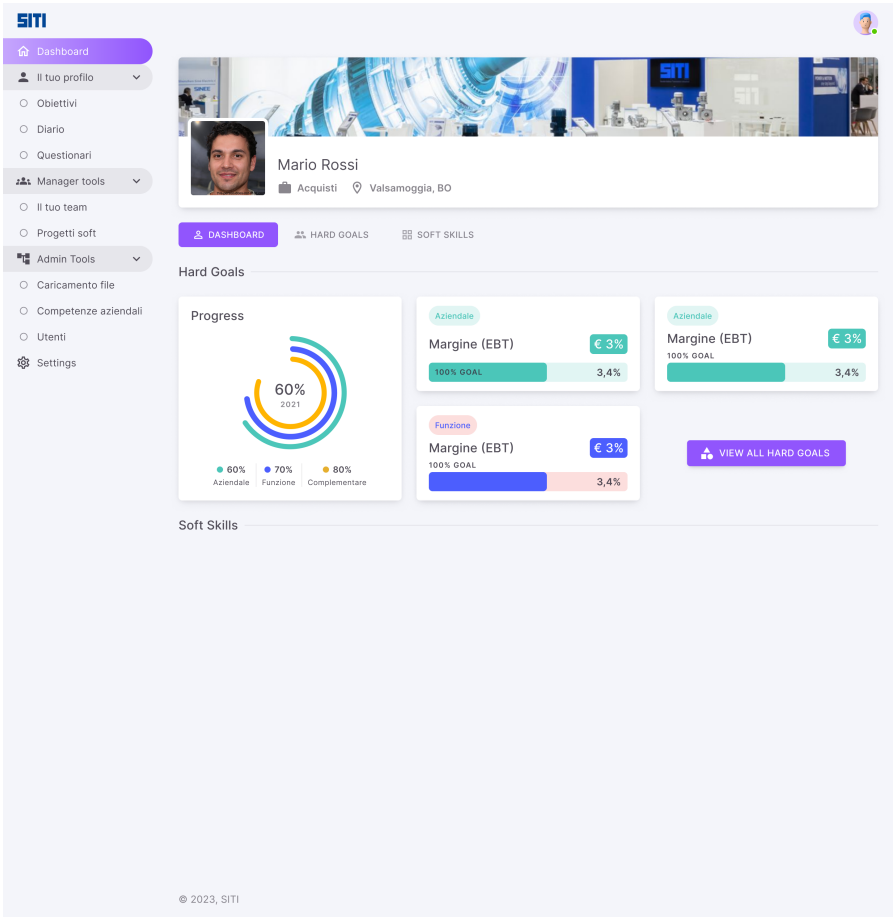

Figure 2: Dipendente - dashboard dei propri obiettivi

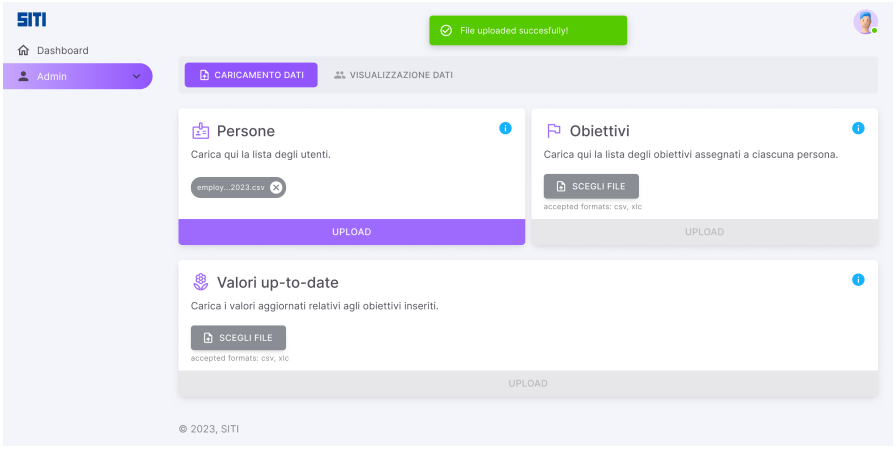

Figure 3: Admin - upload dei dati in formato csv

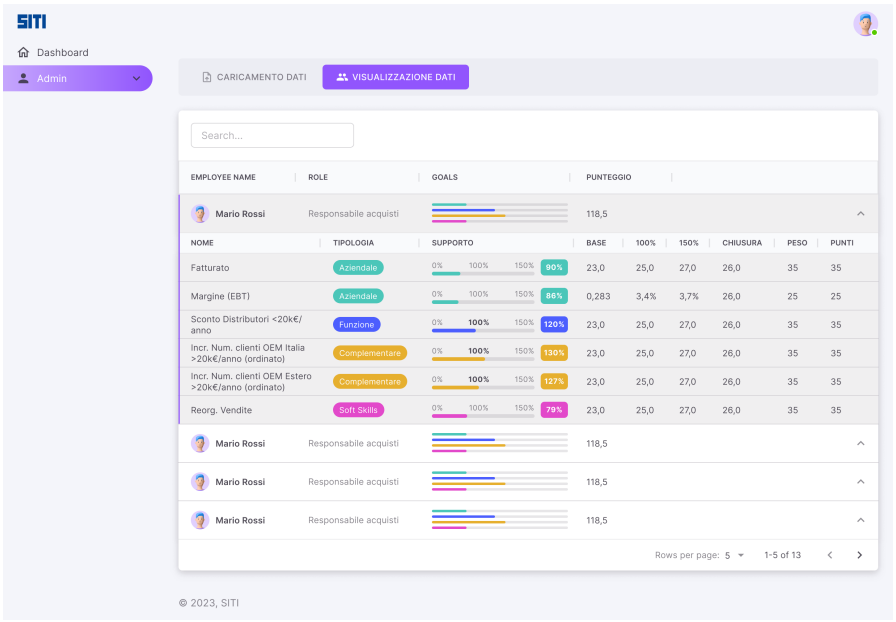

Figure 4: Admin - visione d'insieme su tutti i dipendenti

inseriti. Nei mockup non era stata prevista la colonna up-to-date. Il design finale dell'app consiste in:

- l'admin vede i valori up-to-date di ciascun dipendente
- i dipendenti nella propria dashboard, oltre alla colonna up-to-date, hanno a disposizione anche la colonna "chiusura", che è modificabile e consente di simulare valori possibili di chiusura e vedere come cambierebbe il punteggio.

Architettura

Tecnologie

Nella seguente sezione verranno analizzate le principali tecnologie utilizzate per lo sviluppo del progetto. Per ogni tecnologia verrà fornita una breve descrizione e verranno elencati i vantaggi e gli svantaggi che hanno portato alla scelta o meno di utilizzarla. La scelta delle tecnologie è stata fatta tenendo conto delle conoscenze pregresse dei membri del gruppo e della loro disponibilità a imparare nuove tecnologie, così come della loro popolarità e della loro adozione in ambito aziendale. Inoltre, le scelte dello stack tecnologico hanno tenuto conto dei requisiti che ambo gli stakeholders (ricordiamo la dualità di questo progetto, università e azienda) hanno espresso.

TypeScript

Descrizione

TypeScript è un linguaggio di programmazione open-source sviluppato da Microsoft. È un super-set di JavaScript, ovvero un linguaggio che estende le funzionalità di un altro linguaggio, in questo caso JavaScript.

Vantaggi

Tra i vantaggi di TypeScript troviamo:

- **TypeScript è JavaScript**: essendo un super-set di JavaScript, TypeScript è compatibile con tutte le librerie JavaScript esistenti;
- **TypeScript è un linguaggio fortemente tipizzato**: questo permette di avere un codice più robusto e di trovare più facilmente errori durante la fase di sviluppo;
- **TypeScript è un linguaggio orientato agli oggetti**: questo permette di avere un codice più modulare e di avere una maggiore astrazione rispetto a JavaScript;

Svantaggi

Tra gli svantaggi di TypeScript troviamo:

- **TypeScript è un linguaggio fortemente tipizzato**: l'altro lato della medaglia è che il programmatore deve dichiarare il tipo di ogni variabile, questo può portare ad un aumento della verbosità del codice;

React

React è una libreria JavaScript ampiamente utilizzata per la creazione di interfacce utente reattive e riutilizzabili.

Vantaggi

Tra i vantaggi di React troviamo:

- **Componenti Riutilizzabili**: React permette la creazione di componenti personalizzabili e riutilizzabili, migliorando l'efficienza dello sviluppo;
- **Ampio ecosistema**: la comunità al supporto è molto ampia con molte risorse e librerie disponibili per estendere le funzionalità dell'applicazione;

Svantaggi

Tra gli svantaggi di React troviamo:

- **Curva di apprendimento**: React è una libreria molto ampia e complessa, questo può portare ad una curva di apprendimento molto ripida;

MUI

MUI è una libreria di componenti React che implementa il Material Design di Google. Questa libreria è stata scelta per la sua semplicità e per la sua popolarità.

Vantaggi

Tra i vantaggi di MUI troviamo:

- **Semplicità**: MUI è una libreria molto semplice da utilizzare e da configurare;
- **Popolarità**: MUI è una libreria molto popolare, questo permette di trovare facilmente soluzioni ai problemi che si possono incontrare durante lo sviluppo;
- **Documentazione**: MUI ha una documentazione molto chiara e completa;

Svantaggi

Tra gli svantaggi di MUI troviamo:

- **Personalizzazione**: La personalizzazione dei componenti alle volte può essere difficile;

Docker

Docker è una piattaforma open-source che permette di creare, testare e distribuire applicazioni in maniera semplice e veloce. Docker permette di creare dei container, ovvero degli ambienti virtuali isolati, in cui è possibile eseguire le applicazioni. Nel nostro caso è stato utilizzato per creare un container in cui è possibile eseguire l'applicativo sviluppato. Attraverso Docker è stato possibile eseguire il deploy dell'applicativo.

Vantaggi

Tra i vantaggi di Docker troviamo:

- **Semplicità**: Docker permette di creare e gestire i container in maniera molto semplice;
- **Portabilità**: Docker permette di creare dei container che possono essere eseguiti su qualsiasi sistema operativo.

Svantaggi

Tra gli svantaggi di Docker troviamo:

- **Risorse**: Docker utilizza molte risorse del sistema;
- **Curva di apprendimento**: Docker è una tecnologia molto ampia e complessa, questo può portare ad una curva di apprendimento molto ripida.

MongoDB

MongoDB è un database NoSQL flessibile che consente di gestire dati non strutturati o semi-strutturati.

Vantaggi

Tra i vantaggi di MongoDB troviamo:

- **Flessibilità**: MongoDB permette di gestire dati non strutturati o semi-strutturati;
- **Scalabilità**: MongoDB permette di scalare facilmente il database;
- **Agilità di sviluppo**: MongoDB permette di sviluppare applicazioni in maniera molto veloce, soprattutto grazie all'ottima integrazione con **Mongoose**.

Svantaggi

Tra gli svantaggi di MongoDB troviamo:

- **Dati duplicati**: può soffrire di problemi di dati duplicati;
- **Risorse**: MongoDB utilizza molte risorse del sistema, come ad esempio la memoria del sistema.

Mongoose

Mongoose è una libreria di ODM (Object-Document Mapping) per MongoDB e Node.js. Essenzialmente, Mongoose offre un set di strumenti per semplificare l'interazione con MongoDB, un database NoSQL orientato ai documenti, tramite una rappresentazione in stile oggetto dei dati.

Vantaggi

Tra i vantaggi di Mongoose troviamo:

- **Agilità di sviluppo**: Mongoose semplifica notevolmente lo sviluppo di applicazioni Node.js che utilizzano MongoDB come database di back-end. Fornisce una struttura chiara e coerente per definire schemi, gestire la validazione dei dati e creare query.

Svantaggi

Tra gli svantaggi di Mongoose troviamo:

- **Complessità aggiuntiva**:Mongoose aggiunge una certa complessità all'applicazione a causa della necessità di definire schemi e modelli. Questo potrebbe essere eccessivo per progetti molto semplici o prototipi.

Node.js

Node.js è un ambiente di runtime open-source, multi-piattaforma e orientato agli eventi per l'esecuzione di codice JavaScript lato server.

Vantaggi

Tra i vantaggi di Node.js troviamo:

- **Velocità**: Node.js è molto veloce grazie al suo modello di I/O asincrono;
- **Ampio ecosistema**: la comunità al supporto è molto ampia con molte risorse e librerie disponibili per estendere le funzionalità;
- **JavaScript everywhere**: permette di utilizzare lo stesso linguaggio di programmazione (JavaScript) sia nel frontend che nel backend dell'applicazione, semplificando la condivisione del codice e la sincronizzazione tra il lato client e il lato server.

Svantaggi

Tra gli svantaggi di Node.js troviamo:

- **Single threaded**: sebbene sia in grado di gestire molte richieste simultaneamente, Node.js è single threaded, questo può portare ad un aumento del tempo di risposta in caso di richieste molto complesse;
- **Non adatto per calcoli intensivi**: non è la scelta migliore per applicazioni che richiedono calcoli intensivi o operazioni CPU-bound.

YARN

YARN è un package manager per Node.js che permette di gestire le dipendenze di un progetto. È stato scelto per la sua semplicità e per la sua popolarità.

Vantaggi

Tra i vantaggi di YARN troviamo:

- **Semplicità**: YARN è molto semplice da utilizzare;
- **Velocità**: è noto per le sue prestazioni migliorate rispetto a npm;
- **Riproducibilità**: YARN permette di bloccare le versioni delle dipendenze, questo permette di avere un ambiente di sviluppo riproducibile.

Svantaggi

Tra gli svantaggi di YARN troviamo:

- **Dimensioni**: il binario di YARN è molto più grande rispetto a quello di npm;

Axios

Axios è una libreria JavaScript che permette di effettuare richieste HTTP da Node.js o da browser.

Vantaggi

Tra i vantaggi di Axios troviamo:

- **Semplicità d'uso**: Axios fornisce un'API semplice e intuitiva per effettuare richieste HTTP, sia GET che POST, che si integra facilmente con il codice JavaScript esistente;
- **Supporto alle Promise**: utilizza le Promise per la gestione delle richieste asincrone, il che semplifica notevolmente la gestione del flusso di dati e delle operazioni.

Svantaggi

- **Dimensioni**: Axios è una libreria molto grande, questo può portare ad un aumento delle dimensioni del bundle finale;

CASL

CASL è una libreria JavaScript che permette di gestire i permessi in maniera semplice e dichiarativa. È quindi utilizzata per gestire il controllo degli accessi e l'autorizzazione in applicazioni web, attraverso la definzione di regole d'accesso dinamiche e controllare chi ha il permesso di eseguire determinate azioni all'interno dell'applicazione.

Vantaggi

Tra i vantaggi di CASL troviamo:

- **Controllo Fine-Grained degli Accessi**: consente di definire regole di accesso molto precise;
- **integrazione**: si integra facilmente con diverse tecnologie e framework, come ad esempio React e Node.js;
- **Dinamic policies**: permette di definire regole di accesso dinamiche, che possono cambiare a seconda del contesto;

Svantaggi

Tra gli svantaggi di CASL troviamo:

- **Curva di apprendimento**: CASL è una libreria molto ampia e complessa, questo può portare ad una curva di apprendimento molto ripida, soprattutto per chi non è avezzo al controllo degli accessi;

- **Complessità aggiuntiva**: l'integrazione e lo sviluppo di applicazioni che impiegano CASL tende ad essere più complesso.

csv-parser

csv-parser è una libreria JavaScript che permette di leggere e parsare file CSV.

Vantaggi

Tra i vantaggi di csv-parser troviamo:

- **Semplicità**: è progettato per essere molto semplice da utilizzare;
- **Dimensioni**: è una libreria molto piccola.

Svantaggi

Tra gli svantaggi di csv-parser troviamo:

- **Limitato**: è una libreria molto limitata, infatti non contiene un numero elavato di features.

Codice

Test

Deployment

0.1 Backend

0.1.1 Database schemas

Il design del database è stato effettuato sulla base delle richieste del cliente. L'azienda ci ha fornito un file excel contenente tutti i dati che dovevano essere inseriti nel database.

Sulla base di questo file, sono stati creati diversi schemi, ognuno dei quali rappresenta un entità del dominio.

- **User**: contiene i dati degli utenti, ovvero i dipendenti dell'azienda.
- **Credential**: contiene le credenziali degli utenti.
- **Objective**: contiene tutti gli obiettivi che l'azienda si pone.

- **Assigned**: contiene i valori dell'obiettivo, per ogni utente.
- **Default**: contiene i valori di default di ogni obiettivo.
- **Data**: contiene il valore corrente di un utente per un obiettivo.

Di seguito vengono mostrati gli schemi del database creati con Mongoose.

Users

```
const userSchema = new Schema({
    username: { type: String, required: true, unique: true},
    name: { type: String, required: true, },
    surname: { type: String, required: true },
    position: { type: String, required: true },
    manager: { type: String },
});
```

Da notare che l'unico campo non required è il manager. Abbiamo scelto di non renderlo obbligatorio in quanto ci potrebbero essere delle inconsistenze, o anche solo delle complicazioni, a gestire alcune situazioni. Inoltre, dopo aver parlato con gli stakeholder, è emerso che la gestione del suddetto campo non è fondamentale ai fini del progetto.

La chiave primaria di questo schema è il campo *username*, che è unico.

Credentials

```
const credentialsSchema = new Schema({
    username: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    role: { type: String, required: true }
});
```

TODO

Objectives

```
const objectivesSchema = new Schema({
    name: { type: String, required: true, unique: true },
    type: { type: String, required: true },
})
```

Questo schema è molto semplice: per ogni obiettivo si salvano nome e tipo. Il nome dell'obiettivo è la chiave primaria, ed è unico.

Assigned Objectives

```
const assignedObjectives = new Schema({
    user: { type: String, required: true },
    objective: { type: String, required: true },
    baseValue: { type: Number, required: true },
    value100: { type: Number, required: true },
    value150: { type: Number, required: true },
    weight: { type: Number, required: true },
    year: { type: Number, required: true}
})
```

Questo schema contiene i valori per ogni obiettivo di ogni utente. La chiave primaria è composta da *user* e *objective*, in quanto un utente per un determinato obiettivo può avere solo un valore. La semantica dei vari campi (value100, value150, etc..) deriva dai dati forniti dall'azienda.

Default values

```
const defaultsObjectivesSchema = new Schema({
    objective: { type: String, required: true, unique: true },
    defaultBaseValue: { type: Number, required: true },
    default100: { type: Number, required: true },
    default150: { type: Number, required: true },
    defaultWeight: { type: Number, required: true },
})
```

In questo schema, sono salvati i valori di default per ogni obiettivo. La chiave primaria è il nome dell'obiettivo. I valori di default sono stati forniti dall'azienda e sono utilizzati in caso non vengano assegnati dei valori specifici all'utente.

Data

```
const dataSchema = new Schema({
    objective: { type: String, required: true},
    username: { type: String, required: true },
    date: { type: Date, required: true},
    value: { type: Number, required: true },
})
```

Nel Data schema sono contenuti i valori reali di un utente per un obiettivo in una determinata data.

users/put	Aggiunge uno o più utenti al database (gli utenti già esistenti vengono ag
users/get	Recupera un utente dal database (nel caso non venga specificato uno use
users/delete	Rimuove un utente dal database e come manager di altri utenti, insieme
users/update/manager	Aggiorna il manager di un utente.
users/update/position	Aggiorna la posizione di un utente.
objectives/put	Aggiunge uno o più obiettivi al database (gli obiettivi già esistenti vengo
objectives/get	Recupera un obiettivo dal database (nel caso non venga specificato un n
objectives/delete	Rimuove un obiettivo dal database.
assigned/put	Assegna un obiettivo ad un utente, o più obiettivi a più utenti.
assigned/get	Dato un utente, recupera tutti gli obiettivi a lui assegnati.
assigned/deleteAll	Rimuove tutti gli obiettivi assegnati.
data/put	Aggiunge (o aggiorna) i valori per un utente, relativi ad un obiettivo.
data/get	Recupera i valori di un utente per un obiettivo.
data/deleteAll	Rimuove tutti i valori
auth/me	TODO
auth/login	TODO

0.1.2 API

Le API utilizzano Next.js API routes. Questo framework permette di creare delle API RESTful in modo molto semplice.

Di sono mostrate le routes create:

References

[1] Bernhard Anzengruber, Danilo Pianini, Jussi Nieminen, and Alois Ferscha. Predicting social density in mass events to prevent crowd disasters. In *Social Informatics - 5th International Conference, SocInfo 2013, Kyoto, Japan, November 25-27, 2013, Proceedings*, pages 206–215, 2013.

[2] Giorgio Audrito, Jacob Beal, Ferruccio Damiani, Danilo Pianini, and Mirko Viroli. The share operator for field-based coordination. In *Lecture Notes in Computer Science*, pages 54–71. Springer International Publishing, 2019.

[3] Giorgio Audrito, Mirko Viroli, Ferruccio Damiani, Danilo Pianini, and Jacob Beal. A higher-order calculus of computational fields. *ACM Transactions on Computational Logic*, 20(1):1–55, jan 2019.

[4] Giorgio Audrito, Mirko Viroli, Ferruccio Damiani, Danilo Pianini, and Jacob Beal. On a higher-order calculus of computational fields. In *Formal Techniques*

for Distributed Objects, Components, and Systems, pages 289–292. Springer International Publishing, 2019.

[5] Jacob Beal, Danilo Pianini, and Mirko Viroli. Aggregate programming for the internet of things. *IEEE Computer*, 48(9):22–30, 2015.

[6] Jacob Beal, Mirko Viroli, Danilo Pianini, and Ferruccio Damiani. Self-adaptation to device distribution changes. In *10th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2016, Augsburg, Germany, September 12-16, 2016*, pages 60–69, 2016.

[7] Jacob Beal, Mirko Viroli, Danilo Pianini, and Ferruccio Damiani. Self-adaptation to device distribution in the internet of things. *ACM Trans. Auton. Adapt. Syst.*, 12(3):12:1–12:29, September 2017.

[8] Roberto Casadei, Giancarlo Fortino, Danilo Pianini, Wilma Russo, Claudio Savaglio, and Mirko Viroli. Modelling and simulation of opportunistic IoT services with aggregate computing. *Future Generation Computer Systems*, sep 2018.

[9] Roberto Casadei, Giancarlo Fortino, Danilo Pianini, Wilma Russo, Claudio Savaglio, and Mirko Viroli. A development approach for collective opportunistic edge-of-things services. *Information Sciences*, May 2019.

[10] Roberto Casadei, Danilo Pianini, Guido Salvaneschi, and Mirko Viroli. On context-orientation in aggregate programming. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. IEEE, June 2019.

[11] Roberto Casadei, Danilo Pianini, Mirko Viroli, and Antonio Natali. Self-organising coordination regions: A pattern for edge computing. In *Lecture Notes in Computer Science*, pages 182–199. Springer International Publishing, 2019.

[12] Roberto Casadei, Mirko Viroli, Giorgio Audrito, Danilo Pianini, and Ferruccio Damiani. Aggregate processes in field calculus. In *Lecture Notes in Computer Science*, pages 200–217. Springer International Publishing, 2019.

[13] Pierluigi Contucci, Andrea Omicini, Danilo Pianini, and Alina Sîrbu, editors. *The Future of Digital Democracy*. Springer International Publishing, 2019.

[14] Ferruccio Damiani, Mirko Viroli, Danilo Pianini, and Jacob Beal. Code mobility meets self-organisation: A higher-order calculus of computational fields. In *Formal Techniques for Distributed Objects, Components, and Systems -*

35th IFIP WG 6.1 International Conference, FORTE 2015, Held as Part of the 10th International Federated Conference on Distributed Computing Techniques, DisCoTec 2015, Grenoble, France, June 2-4, 2015, Proceedings, pages 113–128, 2015.

[15] Simon Dobson, Mirko Viroli, Jose Luis Fernandez-Marquez, Franco Zambonelli, Graeme Stevenson, Giovanna Di Marzo Serugendo, Sara Montagna, Danilo Pianini, Juan Ye, Gabriella Castelli, and et al. Spatial awareness in pervasive ecosystems. *The Knowledge Engineering Review*, 31(4):343–366, Sep 2016.

[16] Matteo Francia, Danilo Pianini, Jacob Beal, and Mirko Viroli. Towards a foundational API for resilient distributed systems design. In *2nd IEEE International Workshops on Foundations and Applications of Self* Systems, FAS*W@SASO/ICCAC 2017, Tucson, AZ, USA, September 18-22, 2017*, pages 27–32, 2017.

[17] Sara Montagna, Andrea Omicini, and Danilo Pianini. Extending the gillespie's stochastic simulation algorithm for integrating discrete-event and multi-agent based simulation. In *Multi-Agent-Based Simulation XVI - International Workshop, MABS 2015, Istanbul, Turkey, May 5, 2015, Revised Selected Papers*, pages 3–18, 2015.

[18] Sara Montagna, Andrea Omicini, and Danilo Pianini. A gillespie-based computational model for integrating event-driven and multi-agent based simulation: Extended abstract. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1763–1764, 2015.

[19] Sara Montagna, Danilo Pianini, and Mirko Viroli. Gradient-based self-organisation patterns of anticipative adaptation. In *Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2012, Lyon, France, September 10-14, 2012*, pages 169–174, 2012.

[20] Sara Montagna, Danilo Pianini, and Mirko Viroli. A model for drosophila melanogaster development from a single cell to stripe pattern formation. In *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*, pages 1406–1412, 2012.

[21] Sara Montagna, Mirko Viroli, Matteo Risoldi, Danilo Pianini, and Giovanna Di Marzo Serugendo. Self-organising pervasive ecosystems: A crowd evacuation example. In *Software Engineering for Resilient Systems - Third International Workshop, SERENE 2011, Geneva, Switzerland, September 29-30, 2011. Proceedings*, pages 115–129, 2011.

[22] Danilo Pianini, Jacob Beal, and Mirko Viroli. Improving gossip dynamics through overlapping replicates. In *Coordination Models and Languages - 18th IFIP WG 6.1 International Conference, COORDINATION 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings*, pages 192–207, 2016.

[23] Danilo Pianini, Jacob Beal, and Mirko Viroli. Practical aggregate programming with protelis. In *2nd IEEE International Workshops on Foundations and Applications of Self* Systems, FAS*W@SASO/ICCAC 2017, Tucson, AZ, USA, September 18-22, 2017*, pages 391–392, 2017.

[24] Danilo Pianini, Roberto Casadei, and Mirko Viroli. Security in collective adaptive systems: A roadmap. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. IEEE, June 2019.

[25] Danilo Pianini, Giovanni Ciatto, Roberto Casadei, Stefano Mariani, Mirko Viroli, and Andrea Omicini. Transparent protection of aggregate computations from byzantine behaviours via blockchain. In *Proceedings of the 4th EAI International Conference on Smart Objects and Technologies for Social Good - Goodtechs 18*. ACM Press, 2018.

[26] Danilo Pianini, Angelo Croatti, Alessandro Ricci, and Mirko Viroli. Computational fields meet augmented reality: Perspectives and challenges. In *2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASO Workshops 2015, Cambridge, MA, USA, September 21-25, 2015*, pages 80–85, 2015.

[27] Danilo Pianini, Simon Dobson, and Mirko Viroli. Self-stabilising target counting in wireless sensor networks using euler integration. In *11th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2017, Tucson, AZ, USA, September 18-22, 2017*, pages 11–20, 2017.

[28] Danilo Pianini, Ahmed Elzanaty, Andrea Giorgetti, and Marco Chiani. Emerging distributed programming paradigm for cyber-physical systems over LoRaWANs. In *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE, December 2018.

[29] Danilo Pianini, Sara Montagna, and Mirko Viroli. A chemical inspired simulation framework for pervasive services ecosystems. In *Federated Conference on Computer Science and Information Systems - FedCSIS 2011, Szczecin, Poland, 18-21 September 2011, Proceedings*, pages 667–674, 2011.

[30] Danilo Pianini, Sara Montagna, and Mirko Viroli. Chemical-oriented simulation of computational systems with ALCHEMIST. *J. Simulation*, 7(3):202–215, 2013.

[31] Danilo Pianini and Andrea Omicini. Democratic process and digital platforms: An engineering perspective. In *The Future of Digital Democracy*, pages 83–96. Springer International Publishing, dec 2018.

[32] Danilo Pianini, Stefano Sebastio, and Andrea Vandin. Distributed statistical analysis of complex systems modeled through a chemical metaphor. In *International Conference on High Performance Computing & Simulation, HPCS 2014, Bologna, Italy, 21-25 July, 2014*, pages 416–423, 2014.

[33] Danilo Pianini, Mirko Viroli, and Jacob Beal. Protelis: practical aggregate programming. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pages 1846–1853, 2015.

[34] Danilo Pianini, Mirko Viroli, and Sara Montagna. A simulation framework for pervasive services ecosystems. In *Proceedings of the 12th Workshop on Objects and Agents, Rende (CS), Italy, Jul 4-6, 2011*, pages 150–157, 2011.

[35] Danilo Pianini, Mirko Viroli, Franco Zambonelli, and Alois Ferscha. HPC from a self-organisation perspective: The case of crowd steering at the urban scale. In *International Conference on High Performance Computing & Simulation, HPCS 2014, Bologna, Italy, 21-25 July, 2014*, pages 460–467, 2014.

[36] Danilo Pianini, Sascia Virruso, Ronaldo Menezes, Andrea Omicini, and Mirko Viroli. Self organization in coordination systems using a wordnet-based ontology. In *Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2010, Budapest, Hungary, 27 September - 1 October 2010*, pages 114–123, 2010.

[37] Alessandro Ricci, Mirko Viroli, Andrea Omicini, Stefano Mariani, Angelo Croatti, and Danilo Pianini. *Spatial Tuples: Augmenting Physical Reality with Tuple Spaces*, pages 121–130. Springer International Publishing, Cham, 2017.

[38] Alessandro Ricci, Mirko Viroli, Andrea Omicini, Stefano Mariani, Angelo Croatti, and Danilo Pianini. Spatial tuples: Augmenting reality with tuples. *Expert Systems*, page e12273, apr 2018.

[39] Graeme Stevenson, Juan Ye, Simon Dobson, Danilo Pianini, Sara Montagna, and Mirko Viroli. Combining self-organisation, context-awareness and semantic reasoning: the case of resource discovery in opportunistic networks. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*, pages 1369–1376, 2013.

[40] Mirko Viroli, Giorgio Audrito, Jacob Beal, Ferruccio Damiani, and Danilo Pianini. Engineering resilient collective adaptive systems by self-stabilisation. *ACM Transactions on Modeling and Computer Simulation*, 28(2):1–28, mar 2018.

[41] Mirko Viroli, Giorgio Audrito, Ferruccio Damiani, Danilo Pianini, and Jacob Beal. A higher-order calculus of computational fields. *CoRR*, abs/1610.08116, 2016.

[42] Mirko Viroli, Jacob Beal, Ferruccio Damiani, Giorgio Audrito, Roberto Casadei, and Danilo Pianini. From field-based coordination to aggregate computing. In *Lecture Notes in Computer Science*, pages 252–279. Springer International Publishing, 2018.

[43] Mirko Viroli, Jacob Beal, Ferruccio Damiani, Giorgio Audrito, Roberto Casadei, and Danilo Pianini. From distributed coordination to field calculus and aggregate computing. *Journal of Logical and Algebraic Methods in Programming*, 109:100486, December 2019.

[44] Mirko Viroli, Jacob Beal, Ferruccio Damiani, and Danilo Pianini. Efficient engineering of complex self-organising systems by self-stabilising fields. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems, Cambridge, MA, USA, September 21-25, 2015*, pages 81–90, 2015.

[45] Mirko Viroli, Antonio Bucchiarone, Danilo Pianini, and Jacob Beal. Combining self-organisation and autonomic computing in CASs with aggregate-MAPE. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. IEEE, sep 2016.

[46] Mirko Viroli, Roberto Casadei, and Danilo Pianini. On execution platforms for large-scale aggregate computing. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp Adjunct 2016, Heidelberg, Germany, September 12-16, 2016*, pages 1321–1326, 2016.

[47] Mirko Viroli, Roberto Casadei, and Danilo Pianini. Simulating large-scale aggregate mass with alchemist and scala. In *Proceedings of the 2016 Federated*

Conference on Computer Science and Information Systems, FedCSIS 2016, Gdańsk, Poland, September 11-14, 2016., pages 1495–1504, 2016.

[48] Mirko Viroli, Danilo Pianini, and Jacob Beal. Linda in space-time: An adaptive coordination model for mobile ad-hoc environments. In *Coordination Models and Languages - 14th International Conference, COORDINATION 2012, Stockholm, Sweden, June 14-15, 2012. Proceedings*, pages 212–229, 2012.

[49] Mirko Viroli, Danilo Pianini, Sara Montagna, and Graeme Stevenson. Pervasive ecosystems: a coordination model based on semantic chemistry. In *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*, pages 295–302, 2012.

[50] Mirko Viroli, Danilo Pianini, Sara Montagna, Graeme Stevenson, and Franco Zambonelli. A coordination model of pervasive service ecosystems. *Sci. Comput. Program.*, 110:3–22, 2015.

[51] Mirko Viroli, Danilo Pianini, Alessandro Ricci, Pietro Brunetti, and Angelo Croatti. Multi-agent systems meet aggregate programming: Towards a notion of aggregate plan. In *PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings*, pages 49–64, 2015.

[52] Mirko Viroli, Danilo Pianini, Alessandro Ricci, and Angelo Croatti. Aggregate plans for multiagent systems. *International Journal of Agent-Oriented Software Engineering*, 5(4):336, 2017.

[53] Franco Zambonelli, Andrea Omicini, Bernhard Anzengruber, Gabriella Castelli, Francesco L. De Angelis, Giovanna Di Marzo Serugendo, Simon A. Dobson, Jose Luis Fernandez-Marquez, Alois Ferscha, Marco Mamei, Stefano Mariani, Ambra Molesini, Sara Montagna, Jussi Nieminen, Danilo Pianini, Matteo Risoldi, Alberto Rosi, Graeme Stevenson, Mirko Viroli, and Juan Ye. Developing pervasive multi-agent systems with nature-inspired coordination. *Pervasive and Mobile Computing*, 17:236–252, 2015.