

Unievent

Progetto Cloud Computing

2024

Antonio Lodato



UNIVERSITÀ DEGLI STUDI DI SALERNO

“

E' una piattaforma che consente di pubblicare contenuti. I contenuti possono essere topics o eventi, i contenuti posso avere un immagine descrittiva oppure un video e permettono di connettere persone con gli stessi interessi, attraverso le interazioni e i like. La piattaforma ha tre tipo di utenti: Creator, Organizzatore e Artista. Gli Artisti possono creare sia eventi che topics, i creator possono creare soltanto topics, mentre gli organizzatori solo eventi. Per gli eventi è possibile acquistare un biglietto, ed inoltre è possibile lasciare una recensione.

Funzionalità



Registrazione e Login

L'utente può registrarsi, effettuare il login ed il logout.



Creazione di eventi

L'artista o l'organizzatore può creare un evento, inserendo un titolo, una descrizione e allegare un video o un'immagine.



Creazione di topic

L'utente può creare un topic riguardante uno specifico argomento, inserendo una descrizione e allegare un video o un'immagine.



Chat realtime

Gli utenti possono scambiarsi messaggi con i loro amici.



Visualizzazione di eventi

L'utente può visualizzare, aggiungere ai preferiti, commentare, lasciare like e acquistare biglietti per gli eventi.



Visualizzazione di topic

L'utente può visualizzare, aggiungere ai preferiti, commentare, e lasciare like ai topic.

Funzionalità



Area personale

L'utente può visualizzare i propri dati personali e modificarli. L'utente avrà a disposizione anche un'area per gestire i propri contenuti.



Visualizzare profilo utenti

L'utente può visualizzare il profilo di altri utenti e interagire con i contenuti pubblicati.



Follow

L'utente può seguire ed essere seguito da altri utente, in modo da restare sempre connessi e aggiornati sui contenuti pubblicati.



Verifica Artista

L'utente se riconosciuto dalla legge come artista, può procedere alla verifica artista, attraverso una sezione dedicata «Verifica Account»



Impostazioni

L'utente può settare le proprie preferenze riguardanti le varie sezioni del sito.

Perché Cloud Computing?

- Estendibilità e configurazione agevole dei servizi.
- Velocità di deployment.
- Contenimento dei costi.



Tecnologie utilizzate



NodeJs

Un framework per applicazioni in Javascript lato server basata su un modello di I/O asincrono che opera sugli eventi



MongoDB

E' un database NoSQL orientato ai documenti che memorizza i dati in documenti simili a JSON, chiamati BSON (Binary JSON).



Bootstrap

Framework CSS per lo sviluppo di interfacce Web

Tecnologie utilizzate



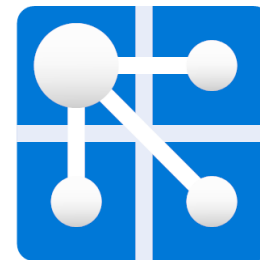
Azure Blob Storage

Azure Blob Storage è un servizio di archiviazione di oggetti di Microsoft Azure progettato per archiviare grandi quantità di dati non strutturati, come file di testo o dati binari.



Azure Functions

Azure Functions è un servizio di calcolo serverless offerto da Microsoft Azure. Permette di eseguire frammenti di codice in risposta a eventi o trigger senza dover gestire l'infrastruttura del server.



Azure Web PubSub

Azure Web PubSub è un servizio di messaggistica in tempo reale completamente gestito che consente di creare applicazioni web e mobili con funzionalità di messaggistica in tempo reale

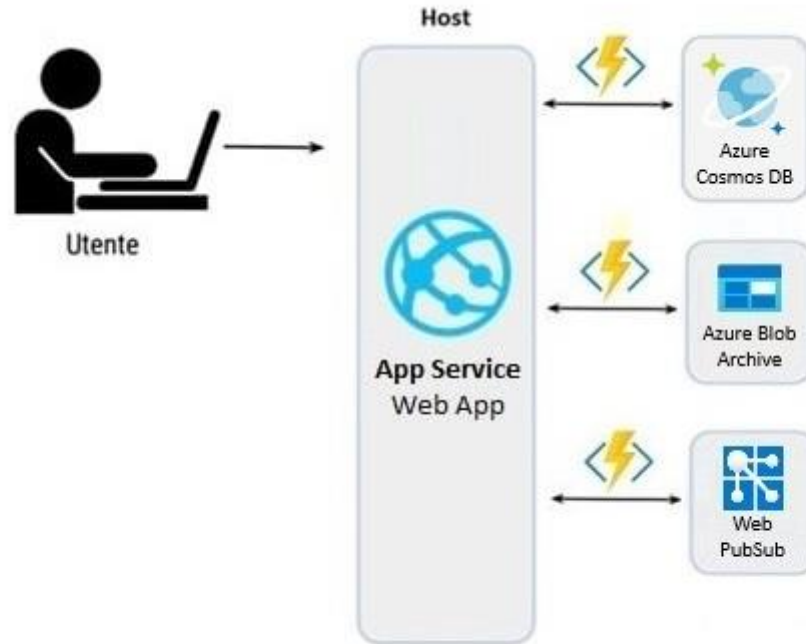
Tecnologie utilizzate



Azure Cosmos DB

Azure Cosmos DB è un database NoSQL completamente gestito che offre una scalabilità globale, una latenza minima e una disponibilità elevata.

Architettura



Azure Bot Service

- Azure Bot Service permette la costruzione di bot intelligenti i quali possono gestire e accedere a molteplici tipologie di dati. Il servizio consente di creare sofisticati assistenti virtuali grazie alla possibilità di sfruttare i servizi offerti da Microsoft Azure per aumentare le proprie funzionalità.
- Attraverso strumenti e librerie open-source e SDK il bot può essere facilmente connesso ai canali di comunicazione più usati come Microsoft Teams, Telegram, Discord, o anche attraverso il canale Direct Line per includerlo in qualsiasi applicazione che implementa la Direct Line API, in modo da essere accessibile da molteplici device.



Azure Bot Service

- Mediante l'integrazione con la raccolta di servizi Azure Cognitive Services è possibile:
 - costruire bot in grado di parlare
 - ascoltare e comprendere l'utente che è così in grado di comunicare in linguaggio naturale con esso.
- Tale integrazione risulta semplice da effettuare grazie alla interconnessione fornita direttamente dall'Azure Bot Service.



Strumenti per lo sviluppo

- Il ciclo di vita di un bot segue uno sviluppo simile a quello previsto per qualsiasi altro applicativo.
- Bot Framework è una raccolta di tool utilizzati per creare, testare, distribuire e gestire bot intelligenti integrandoli con i diversi servizi Azure, il tutto in un unico ambiente. Comprende:
 - SDK per vari linguaggi di programmazione;
 - lo strumento visuale Bot Framework Composer;
 - il simulatore per il test Bot Framework Emulator.



Plan:

Review the bot [design guidelines](#)



Build:

[Download Command Line tool](#)

Create a bot [from Azure](#) or [locally](#)

Add services such as

[Language Understanding \(LU\)](#)
[Dispatch](#)



Test:

Test with the [Emulator](#)

Test online in [Web Chat](#)



Publish:

Publish directly to Azure or

Use [Continuous Deployment](#)



Connect:

Connect to [channels](#)



Evaluate:

[View analytics](#)

Bot App Service

- Il portale di Azure fornisce la possibilità di creare un nuovo bot, utilizzando il pannello di configurazione del servizio.
- Dal portale è possibile inoltre testare il funzionamento del bot mediante un'apposita Web Chat.
- Una volta effettuato il deploy del bot, si ottiene un endpoint di messaggistica il quale verrà utilizzato dal bot di Azure per scambiare messaggi tra l'applicazione e l'app service.


Piano dei prezzi del servi... : S1


Endpoint di messaggistica : <https://botunievent.azurewebsites.net/api/messages>

Crea app Web

Dettagli del progetto

Selezionare una sottoscrizione per gestire le risorse distribuite e i costi. Usare gruppi di risorse, ad esempio le cartelle, per organizzare e gestire tutte le risorse.


Sottoscrizione *  Azure per studenti

Gruppo di risorse *  UniEventBotResources

[Crea nuovo](#)

Dettagli dell'istanza

È necessario un database? [Prova la nuova esperienza Web + database.](#)


Nome * botunievent2  .azurewebsites.net

Pubblica * ☒ Codice ☐ Contenitore Docker

Stack di runtime * Node 14 LTS


Sistema operativo * ☒ Linux ☐ Windows

Area geografica * Australia Central

 Se non si trova il piano di servizio app, provare a selezionare un'altra area.

Piano di servizio app

Il piano tariffario del piano di servizio app determina la località, le funzionalità, i costi e le risorse di calcolo associati all'app. [Altre informazioni](#)

Piano Linux (Australia Central) *  (Nuovo) ASP-UniEventBotResources-b217

[Crea nuovo](#)

SKU e dimensioni * **Gratuito F1**
1 GB di memoria
[Modifica dimensioni](#)

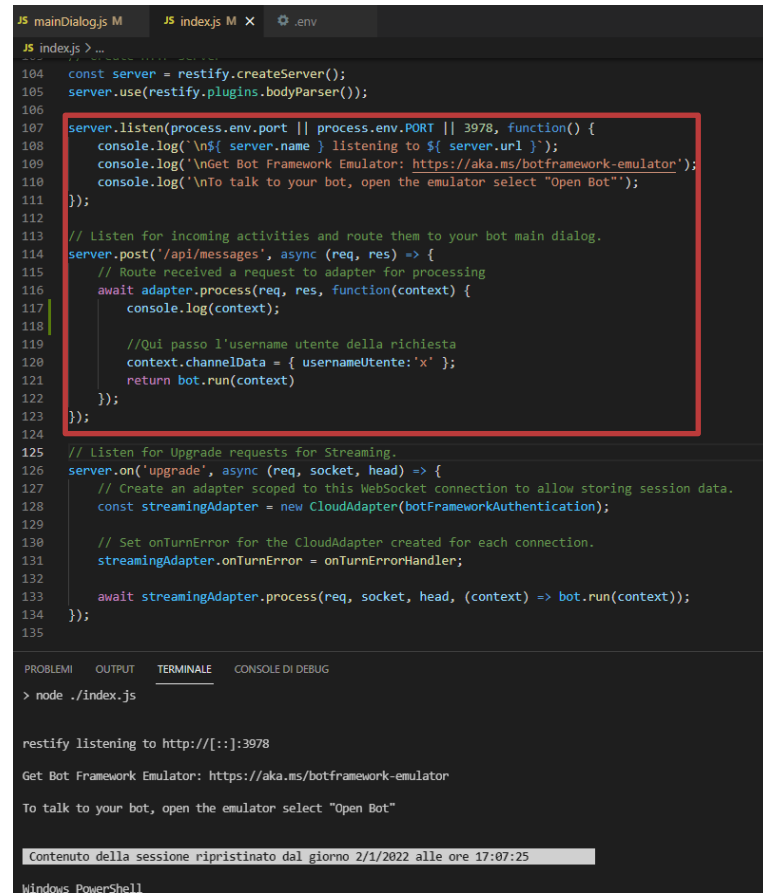
[Rivedi e crea](#)

[< Indietro](#)

[Avanti: Distribuzione >](#)

UniEventBot

- UniEventBot è un bot progettato per fornire supporto a tutti gli utenti dell'applicazione UniEvent.
- Il bot consente all'utente di eseguire azioni rapide all'interno dell'applicazione.
- Le azioni svolte sono:
 - Visualizzazione del nome
 - Modifica di dati personali
 - Aggiunta di nuovi amici
 - Visualizzare la lista di topic ed eventi
 - Ricerca e creazione di eventi e topic
 - Visualizzazione del profilo
- Il codice a lato mostra come il bot viene messo in ascolto sulla porta 3978 in attesa di richieste per poter svolgere azioni.



```
JS mainDialogjs M JS indexjs M X .env
JS indexjs > ...
104 const server = restify.createServer();
105 server.use(restify.plugins.bodyParser());
106
107 server.listen(process.env.port || process.env.PORT || 3978, function() {
108   console.log(`\n${ server.name } listening to ${ server.url }`);
109   console.log(`\nGet Bot Framework Emulator: https://aka.ms/botframework-emulator`);
110   console.log(`\nInfo talk to your bot, open the emulator select "Open Bot"`);
111 });
112
113 // Listen for incoming activities and route them to your bot main dialog.
114 server.post('/api/messages', async (req, res) => {
115   // Route received a request to adapter for processing
116   await adapter.process(req, res, function(context) {
117     console.log(context);
118
119     //Qui passo l'username utente della richiesta
120     context.channelData = { usernameUtente: 'x' };
121     return bot.run(context)
122   });
123 });
124
125 // Listen for Upgrade requests for Streaming.
126 server.on('upgrade', async (req, socket, head) => {
127   // Create an adapter scoped to this WebSocket connection to allow storing session data.
128   const streamingAdapter = new CloudAdapter(botFrameworkAuthentication);
129
130   // Set onTurnError for the CloudAdapter created for each connection.
131   streamingAdapter.onTurnError = onTurnErrorHandler;
132
133   await streamingAdapter.process(req, socket, head, (context) => bot.run(context));
134 });
135
PROBLEMI OUTPUT TERMINALE CONSOLE DI DEBUG
> node ./index.js

restify listening to http://[::]:3978

Get Bot Framework Emulator: https://aka.ms/botframework-emulator

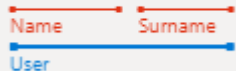
To talk to your bot, open the emulator select "Open Bot"

Contenuto della sessione ripristinato dal giorno 2/1/2022 alle ore 17:07:25
Windows PowerShell
```

UniEventBot

- UniEventBot utilizza LUIS come modello per elaborare le istruzioni inviate dall'utente.
- L'intelligenza artificiale fornita da LUIS è basata sul Natural Language Understanding (NLU), una componente del più complesso Natural Language Processing (NLP). Il NLU interpreta il significato di un testo e lo classifica in specifici intent, che corrispondono alle intenzioni che il testo esprime.
- LUIS utilizza delle Entity "per mappare" le entità contenute all'interno degli Intent che riconosce.

aggiungi autore michele troisi



Intents ⓘ

+ Create + Add prebuilt domain intent ⌵ Rename 🗑 Delete	
<input type="radio"/> Name ↑	Examples
<input type="radio"/> AddNewFriend	10
<input type="radio"/> CreateNewEvent	3
<input type="radio"/> CreateNewTopic	3
<input type="radio"/> EditInfoUser	8
<input type="radio"/> GetEventList	5
<input type="radio"/> GetName	3
<input type="radio"/> GetTopicList	3
<input type="radio"/> None	0
<input type="radio"/> SearchEvent	8
<input type="radio"/> SearchTopic	8

UniEventBot

- Per collegare LUIS al bot si utilizza un file di configurazione .env

```
JS mainDialog.js M .env X
.env
1 MicrosoftAppType= MultiTenant
2 MicrosoftAppId= 965a11ec-d360-4b09-b22b-d1529c9056a8
3 MicrosoftAppPassword= ~ws7Q-3Kr16r1-dLeIX.Kg5hwj-PleQuhoEiT
4 MicrosoftAppTenantId=
5 LuisAppId= 0d519db2-fd29-4c41-a45b-81729ce3f4e
6 LuisAPIKey= 80390e9bf2e54c2789339200ccb0d60f
7 LuisAPIHostName= westeurope.api.cognitive.microsoft.com
```

- La struttura di una conversazione viene definita mediante una serie di oggetti dialog, il quale rappresenta l'unità di base del flusso di esecuzione della conversazione. Ogni dialog ha un inizio, uno svolgimento e una fine, può essere messo in pausa e ripreso o essere cancellato. I dialog possono essere visti come funzioni di un linguaggio di programmazione a cui possono essere passati argomenti o parametri e che alla fine restituiscono un valore. La classe bot è solitamente accompagnata da un dialog principale che poi si ramifica in vari dialog a seconda dello svolgersi della conversazione.

```
class MainDialog extends ComponentDialog {
  constructor(luisRecognizer, bookingDialog, addNewFriendDialog) {
    super('MainDialog');

    if (!luisRecognizer) throw new Error('[MainDialog]: Missing parameter \'luisRecognizer\' is required');
    this.luisRecognizer = luisRecognizer;

    if (!bookingDialog) throw new Error('[MainDialog]: Missing parameter \'bookingDialog\' is required');

    if (!addNewFriendDialog) throw new Error('[MainDialog]: Missing parameter \'addNewFriendDialog\' is required');

    // Define the main dialog and its related components.
    // This is a sample "book a flight" dialog.
    this.addDialog(new TextPrompt('TextPrompt'))
      .addDialog(bookingDialog)
      .addDialog(addNewFriendDialog)
      .addDialog(new WaterfallDialog(MAIN_WATERFALL_DIALOG, [
        this.introStep.bind(this),
        this.actStep.bind(this),
        this.finalStep.bind(this)
      ]));

    this.initialDialogId = MAIN_WATERFALL_DIALOG;
  }

  /**
   * Second step in the waterfall. This will use LUIS to attempt to extract the origin, destination and travel dates.
   * Then, it hands off to the bookingDialog child dialog to collect any remaining details.
   */
  async actStep(stepContext) {
    const friendDetails = {};

    if (!this.luisRecognizer.isConfigured) {
      // LUIS is not configured, we just run the AddNewFriend path.
      return await stepContext.beginDialog('addNewFriendDialog', friendDetails);
    }

    // Call LUIS and gather any potential booking details. (Note the TurnContext has the response to the prompt)
    const luisResult = await this.luisRecognizer.executeLUISQuery(stepContext.context);
    switch (luisRecognizer.topIntent(luisResult)) {

      case 'CreateNewEvent': {
        // We haven't implemented the GetWeatherDialog so we just display a TODO message.
        const getCreateNewEventMessageText = 'Link per creare un nuovo evento: '+ $(baseURL)/home/eventi/addEvento?'\n';
        await stepContext.context.sendActivity(getCreateNewEventMessageText, getCreateNewEventMessageText, InputHints.IgnoreInput);
        break;
      }
    }
  }
}
```


UniEventBot

- Esistono diversi tipi di dialog che permettono di costruire al meglio l'interazione con l'utente.

- Component dialog - la base di ogni conversazione, può contenere diversi dialog al suo interno permettendo di suddividere conversazioni complesse in pezzi più piccoli e possono a loro volta essere contenuti all'interno di bot o di altri dialog.
- Waterfall dialog - definisce una conversazione composta da una serie di passi che ha solitamente lo scopo di ottenere informazioni dall'utente mediante la classe prompt

- LuisRecognizer permette la connessione all'istanza LUIS in esecuzione su cloud. Nella figura viene mostrato il codice che gestisce la creazione un nuovo evento, e restituisce il link che reindirizza l'utente alla pagina di creazione di un nuovo evento.

```
class MainDialog extends ComponentDialog {
  constructor(luisRecognizer, bookingDialog, addNewFriendDialog) {
    super('MainDialog');

    if (!luisRecognizer) throw new Error('[MainDialog]: Missing parameter \'luisRecognizer\' is required');
    this.luisRecognizer = luisRecognizer;

    if (!bookingDialog) throw new Error('[MainDialog]: Missing parameter \'bookingDialog\' is required');

    if (!addNewFriendDialog) throw new Error('[MainDialog]: Missing parameter \'addNewFriendDialog\' is required');

    // Define the main dialog and its related components.
    // This is a sample "book a flight" dialog.
    this.addDialog(new TextPrompt('TextPrompt'))
      .addDialog(bookingDialog)
      .addDialog(addNewFriendDialog)
      .addDialog(new WaterfallDialog(MAIN_WATERFALL_DIALOG, [
        this.introStep.bind(this),
        this.actStep.bind(this),
        this.finalStep.bind(this)
      ]));

    this.initialDialogId = MAIN_WATERFALL_DIALOG;
  }

  /**
   * Second step in the waterfall. This will use LUIS to attempt to extract the origin, destination and travel dates.
   * Then, it hands off to the bookingDialog child dialog to collect any remaining details.
   */
  async actStep(stepContext) {
    const friendDetails = {};

    if (this.luisRecognizer.isConfigured) {
      // LUIS is not configured, we just run the AddNewFriend path.
      return await stepContext.beginDialog('addNewFriendDialog', friendDetails);
    }

    // Call LUIS and gather any potential booking details. (Note the TurnContext has the response to the prompt)
    const luisResult = await this.luisRecognizer.executeLUISQuery(stepContext.context);
    switch (luisRecognizer.topIntent(luisResult)) {
      case 'CreateNewEvent': {
        // We haven't implemented the GetWeatherDialog so we just display a TODO message.
        const getCreateNewEventMessageText = `Link per creare un nuovo evento: "${(baseURL)/home/eventi/addEvento}"`;
        await stepContext.context.sendActivity(getCreateNewEventMessageText, getCreateNewEventMessageText, InputHints.IgnoringInput);
        break;
      }
    }
  }
}
```

Il Team



Antonio Lodato

DEMO

