

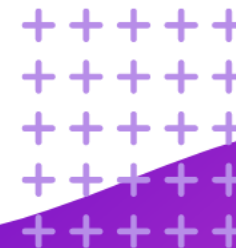
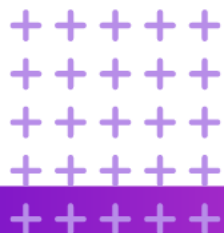


**Blazor**  
Conf 2024



# Real-time application con Blazor

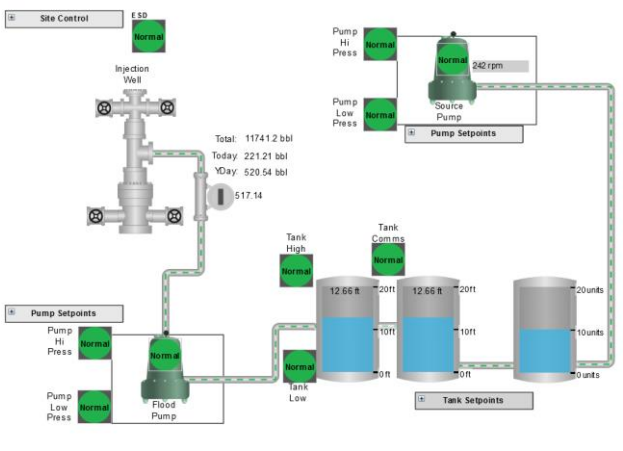
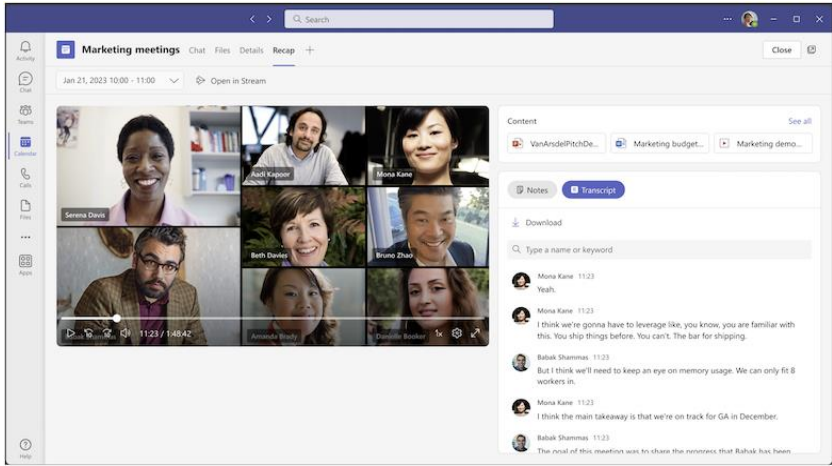
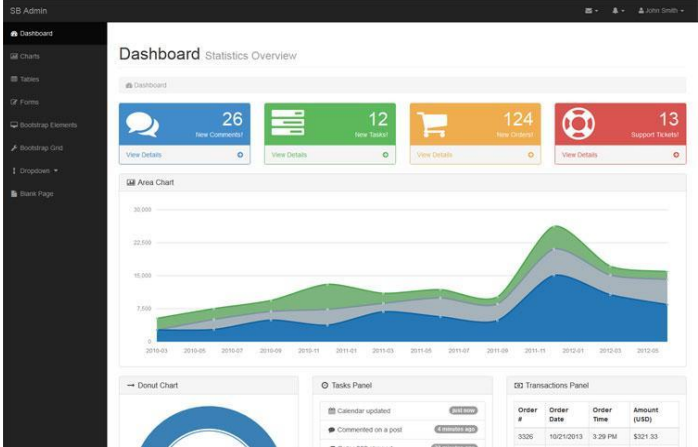
Andrea Dottor



Un grazie agli sponsor



# Real-time application



# polling vs pushing

## Polling (Pull)

- 👎 Consumo maggiore di dati/traffico di rete
- ✓ Di facile implementazione

HTTP-based APIs,  
RESTful services, ...

## Pushing

- 👎 Aumento della complessità
- ✓ Il client viene aggiornato solo se (e quando) è necessario

WebSocket,  
Server Sent Events, ...

# Blazor Web App

InteractiveServer

InteractiveWebAssembly

InteractiveAuto

Possiamo far funzionare componenti sia come Blazor Server che Blazor WebAssembly, e tutto nella stessa applicazione

- @rendermode

# Dependency Injection & Events

- Un servizio dichiarato come *scoped* in Blazor (interactive), equivale ad un *singleton* ma limitato però al singolo utente.
- Con `InteractiveServer`, un servizio dichiarato come *singleton* permette di condividere informazioni (ed eventi) per l'intera applicazione, con tutti gli utenti.
- Gli *eventi* permettono di poter far notificare ad un componente quando abbiamo un aggiornamento dei dati.

# BackgroundService, IHostedService

All'interno dell'applicazione (server) possiamo eseguire del codice in background, utile per sottoscrivere alla ricezione di messaggi da servizi esterni (es. ServiceBus, RabbitMQ, ...)

Gli *Hosted-Service* permettono di avere della logica che viene avviata in automatico allo start dell'applicazione



# Fluent UI

The screenshot shows the Microsoft Fluent UI Blazor components library website. The header features the Microsoft logo, a search bar, and user profile icons. The left sidebar contains navigation links for Home, Getting Started (with sub-links like What's new, Upgrade guide, Code setup, Project templates, Themes, Design tokens, Reboot, Icons and Emoji), Services, Layout, Form & Inputs, and Components (with sub-links like Overview, Accordion, Anchor, Anchored Region, and AppBar). The main content area is titled "Welcome to the Fluent UI Blazor components library" and includes a message about version 4.6.1 supporting .NET 8 only, with links to previous versions (3.6.0) and their supported .NET versions (6 and 7). Below this is an "Introduction" section explaining the `Microsoft.FluentUI.AspNetCore.Components` package and its purpose. The bottom section, "Open at Microsoft - 2024", features three video thumbnails: "Getting started with the Fluent UI Blazor", "Exploring the Fluent UI Blazor components", and "Advanced scenario with FluentUI-Blazor". A fourth video thumbnail for ".NET Conf 2024" is partially visible. The right sidebar contains a table of contents for the article, a console log showing notifications and settings, and a footer with the version number (4.6.1+ff23edc6) and copyright information (Microsoft © 2024).

Microsoft

Search everything...

Home

Getting Started

- What's new
- Upgrade guide
- Code setup
- Project templates
- Themes
- Design tokens
- Reboot
- Icons and Emoji

Services

Layout

Form & Inputs

Components

- Overview
- Accordion
- Anchor
- Anchored Region
- AppBar

## Welcome to the Fluent UI Blazor components library

This is the demo and documentation site for version **4.6.1** of the library. This version supports **.NET 8 only**

The demo and documentation sites for previous version is also still available:  
[Version 3.6.0](#)  
This version supports both .NET 6 and .NET 7.

## Introduction

The `Microsoft.FluentUI.AspNetCore.Components` package provides a set of [Blazor](#) components. Some of the components are wrappers around Microsoft's official Fluent UI Web Components. Others are components that leverage the Fluent Design System or make it easier to work with Fluent. To get up and running with the `Microsoft.FluentUI.AspNetCore.Components` library, see the "Getting Started" section below.

The source for the library is hosted in the [fluentui-blazor](#) repository at GitHub.

### Open at Microsoft - 2024

- Getting started with the Fluent UI Blazor
- Exploring the Fluent UI Blazor components
- Advanced scenario with FluentUI-Blazor
- .NET Conf 2024: Unlocking the power of the Fluent UI Blazor components

### In this article

- Introduction
- What's new?
- Getting Started
  - Scripts
  - Styles
  - Reboot
  - Code setup
- Getting started by using project templates
- Using the Fluent UI Blazor components
  - Simple components example
- Configuring the Design System
- Blazor Hybrid
  - Temporary workaround for MAUI/WPF/Windows Forms issues
- Use the DataGrid component with EF Core
  - Installation
  - Usage
- Support
- Joining the Community

### Console log

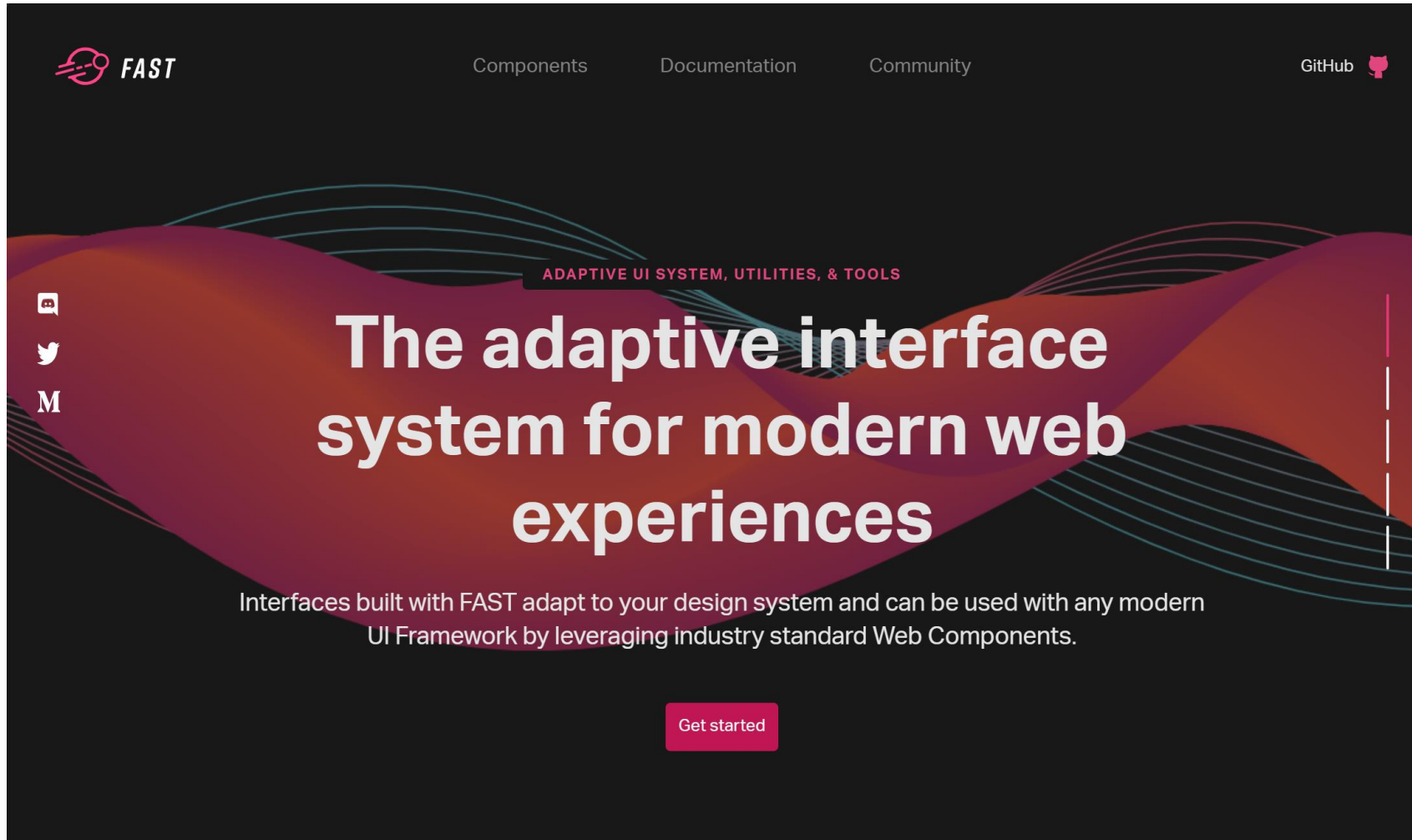
- [12:44:38] - Open notification center
- [12:44:39] - Notification center dismissed
- [12:44:42] - Open site settings

Version: 4.6.1+ff23edc6 - Powered by .NET 8.0.4

Microsoft © 2024. All rights reserved.



# FAST's Adaptive UI technology

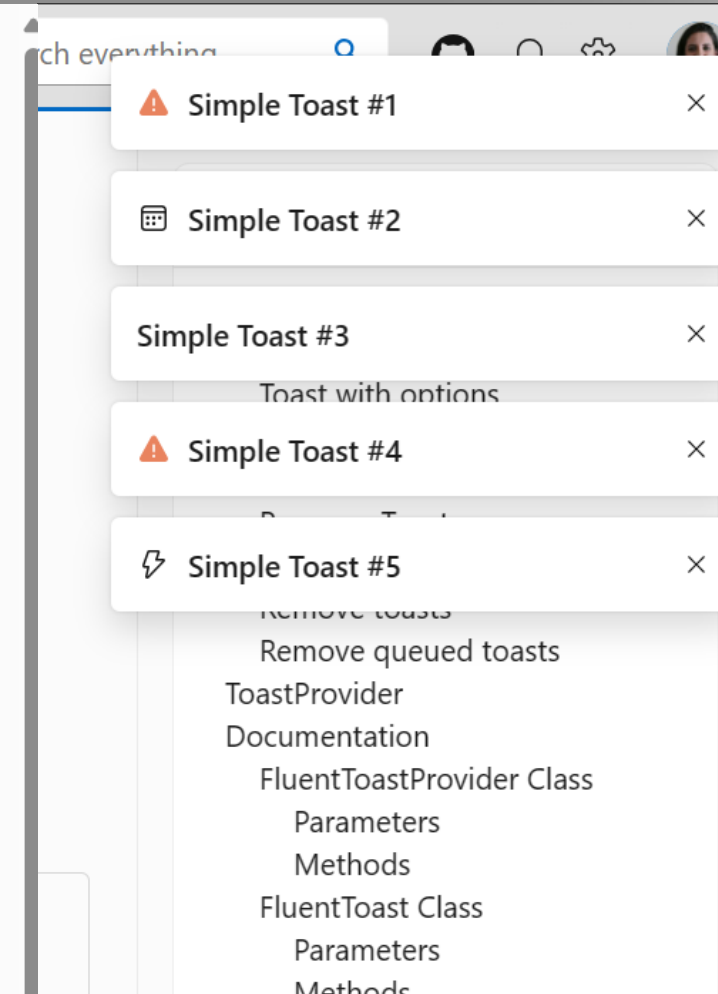
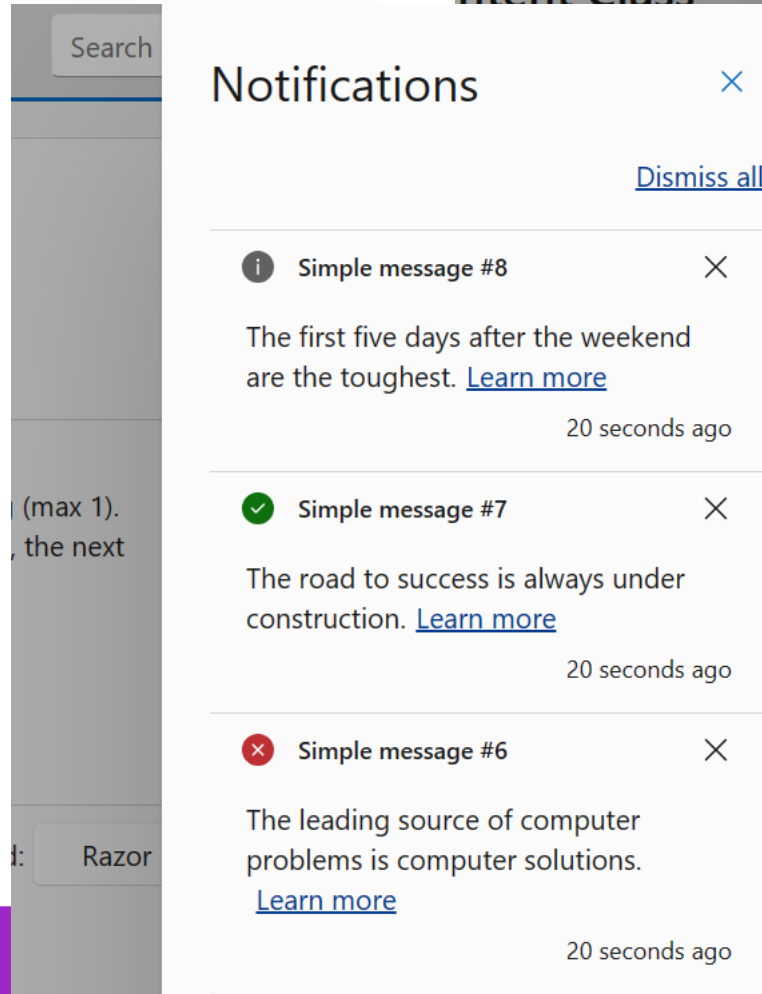
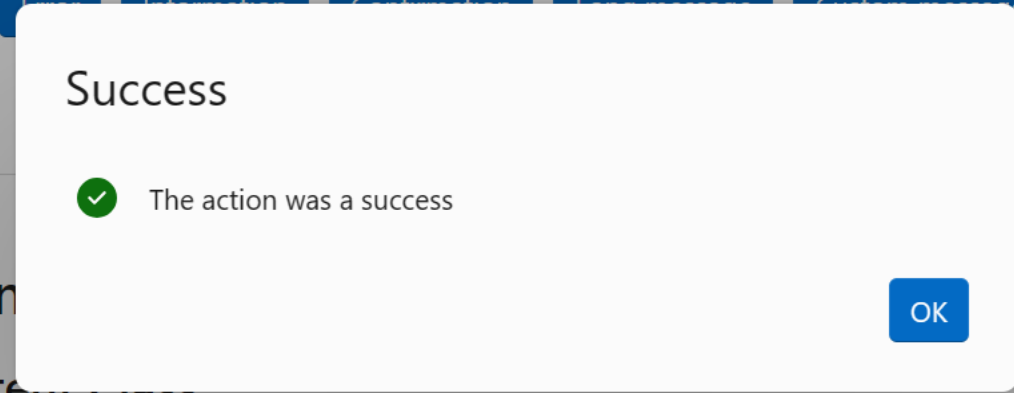


# Fluent UI - Services

DialogService

MessageService

ToastService

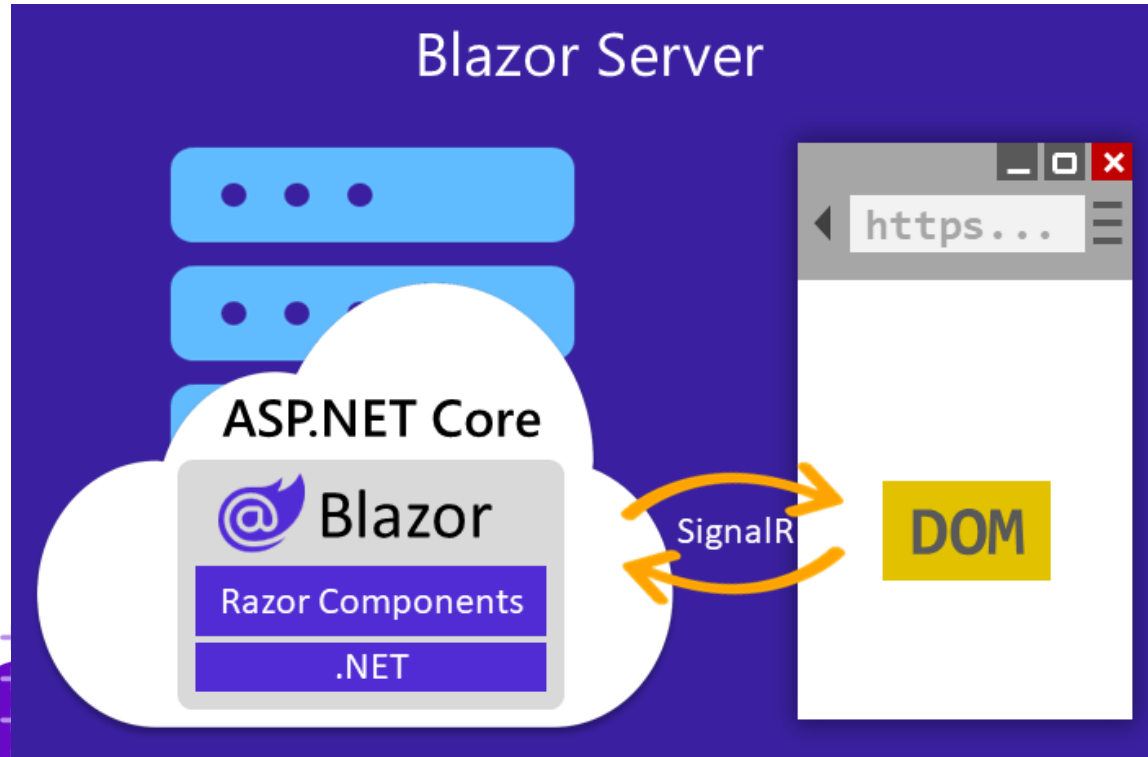


# Interactive Server

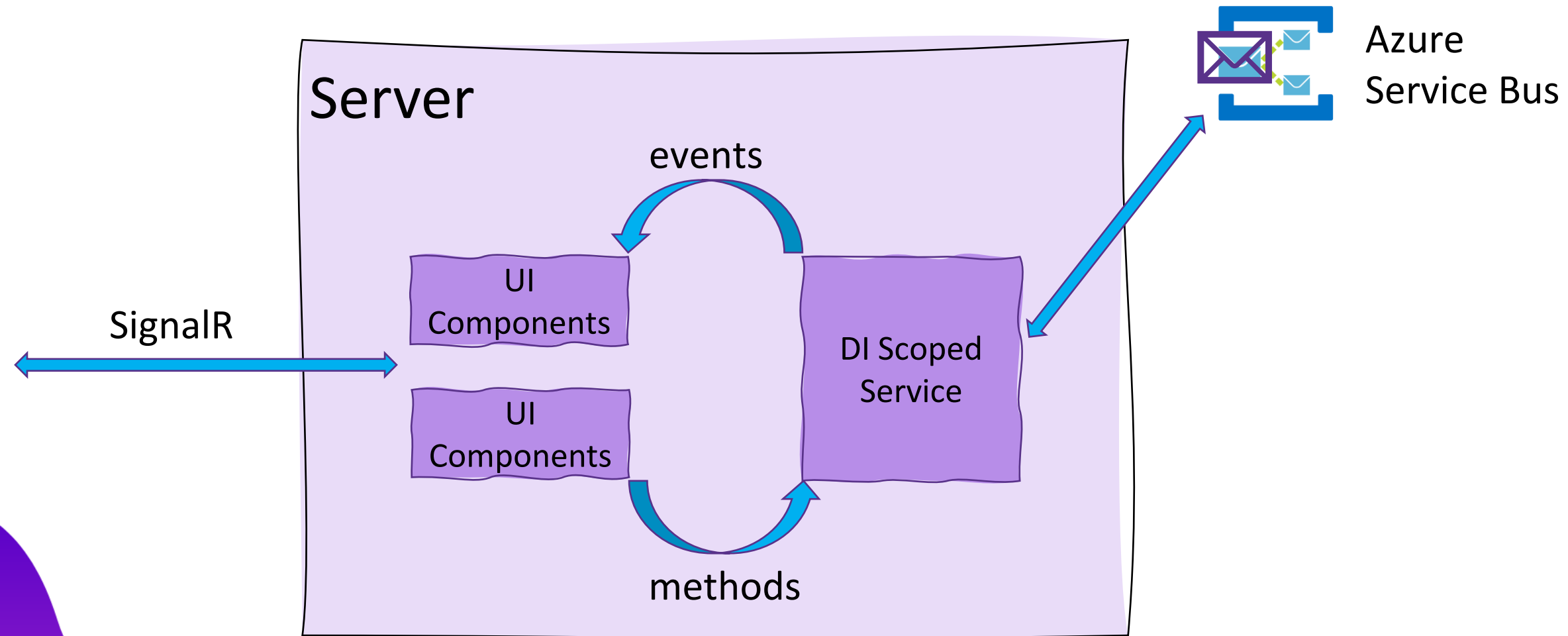


# Interactive Server

- L'applicazione viene eseguita sul server
- Il client è collegato tramite SignalR



# DI Scoped + Events





# Demo

Interactive Server



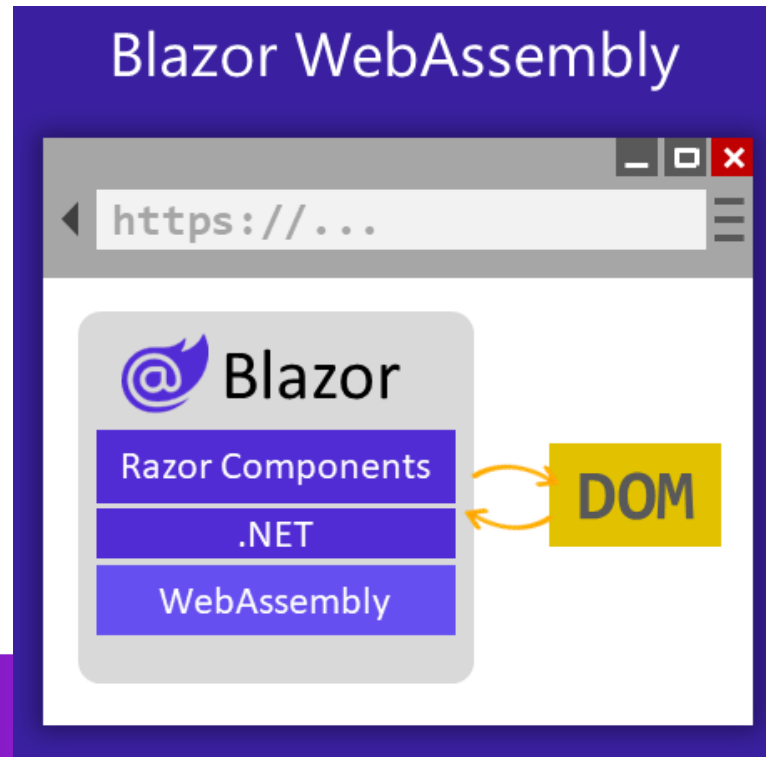


# Interactive WebAssembly



# Interactive WebAssembly

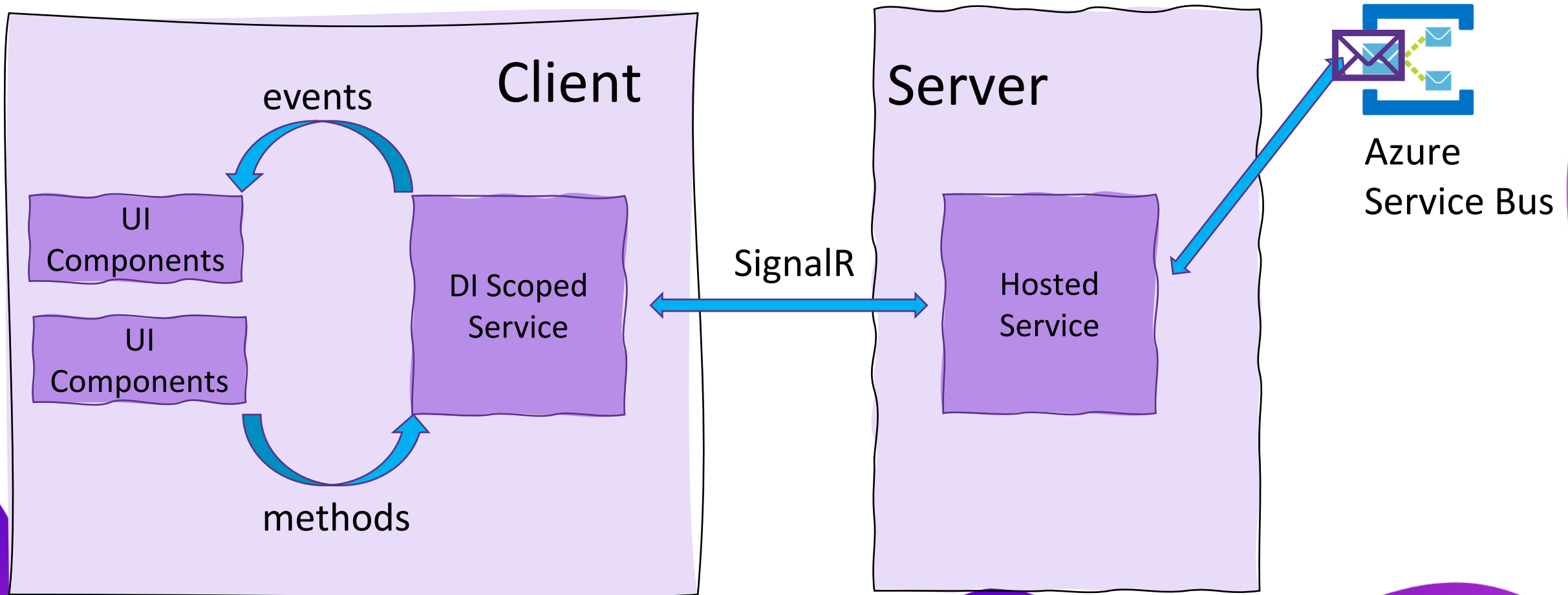
- L'applicazione viene eseguita nel browser
- Possiamo utilizzare SignalR per ricevere gli eventi dal server
  - Sta a noi gestire la connessione
  - Sta a noi gestire correttamente l'invio e la ricezione dei messaggi



# IHostedService + SignalR + DI Scoped + Events

- Un servizio in background può sottoscrivere alla ricezione di eventi da servizi esterni (es. Azure Service Bus, ...)
- Utilizzare SignalR per inviare il messaggio ricevuto a tutti i client che ne hanno bisogno
- Utilizzare un servizio scoped per ricevere i messaggi da SignalR e scatenare l'evento (gestito dai component di UI)

# IHostedService + SignalR + Scoped + Events



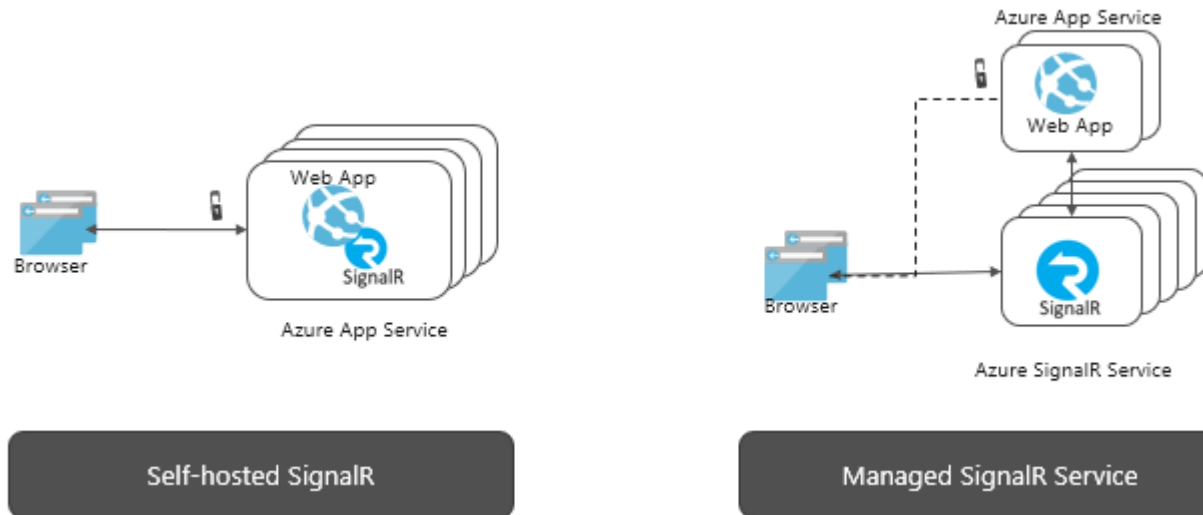


# Demo

Interactive WebAssembly

# Possiamo ottimizzare?

- Se riceviamo una grande quantità di dati, è possibile valutare di sostituire SignalR (verso il server) con servizi esterni all'applicazione.
- Azure SignalR o Azure Web PubSub sono una possibile scelta.



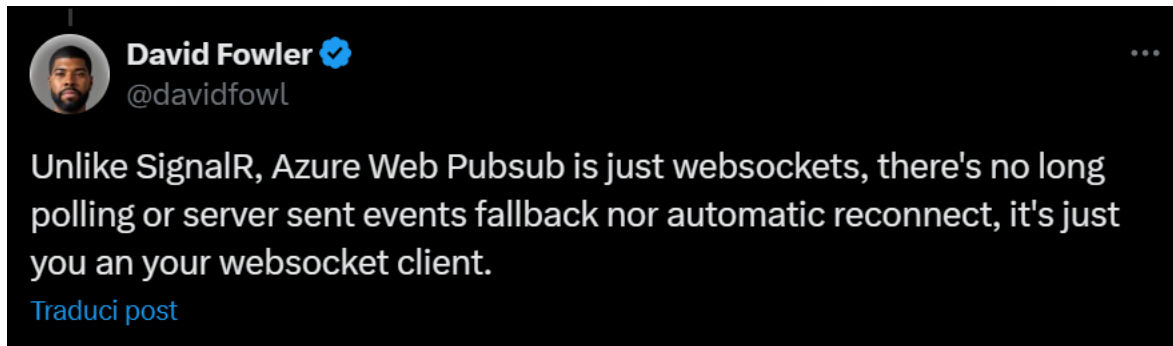
# Azure Web PubSub

## Serverless

Client si collega direttamente ai servizi su Azure, senza bisogno di dover passare per il server

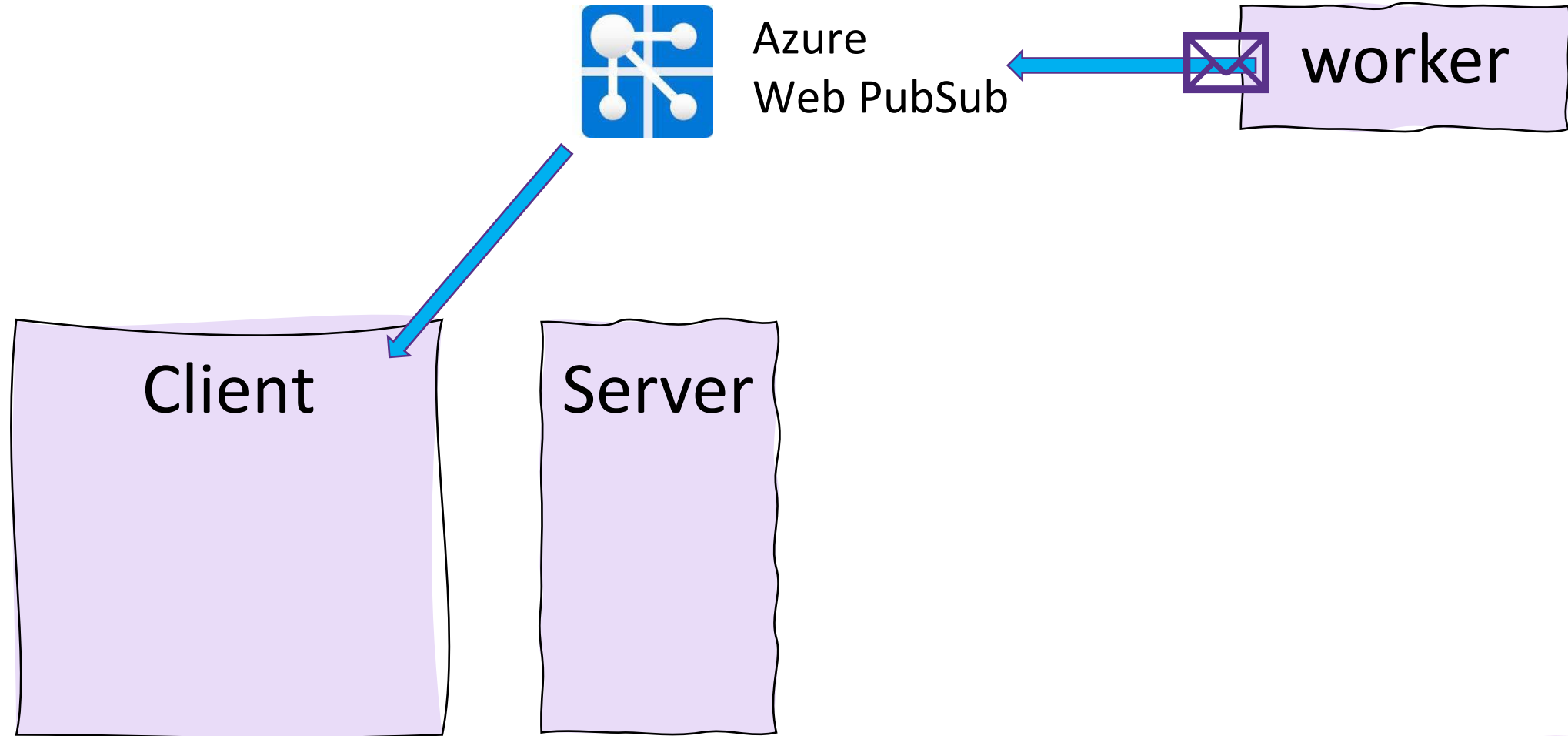
Supporta solo WebSocket. Sta a noi tutta la gestione del protocollo, riconessioni, ...

# Azure Web PubSub vs Azure SignalR





# Push messages from server





# Demo

Posizione gps in real-time



E ora...

- Interactive Server o Interactive WebAssembly?
- SignalR o gRPC?
- Azure Service Bus, Azure SignalR o Azure Web PubSub?

¯\\_ (ツ) \_/¯ dipende...



# Codice e slide su github



- <https://github.com/andreadottor/Umarell-ParentalControl>

# Grazie!



@andreadottor

{

name:

**Andrea Dottor**

email:

**andrea@dottor.net**

x:

**@dottor**

linkedin:

**@andreadottor**

}

