



# Blogq

A new low-code  
platform/framework

Based on Microsoft .NET stack



# Highlights

One powerful programming language: C#

Modeling: both in code and in browser

Just a NuGet package

Based on commodity tech

Deploy anywhere



## Tech stack



BLAZOR WASM



MINIMAL WEB APIS



MONGODB  
(DEFAULT  
DATABASE)



.NET ASPIRE  
(DEPLOYMENT)

## Two Ways to Model

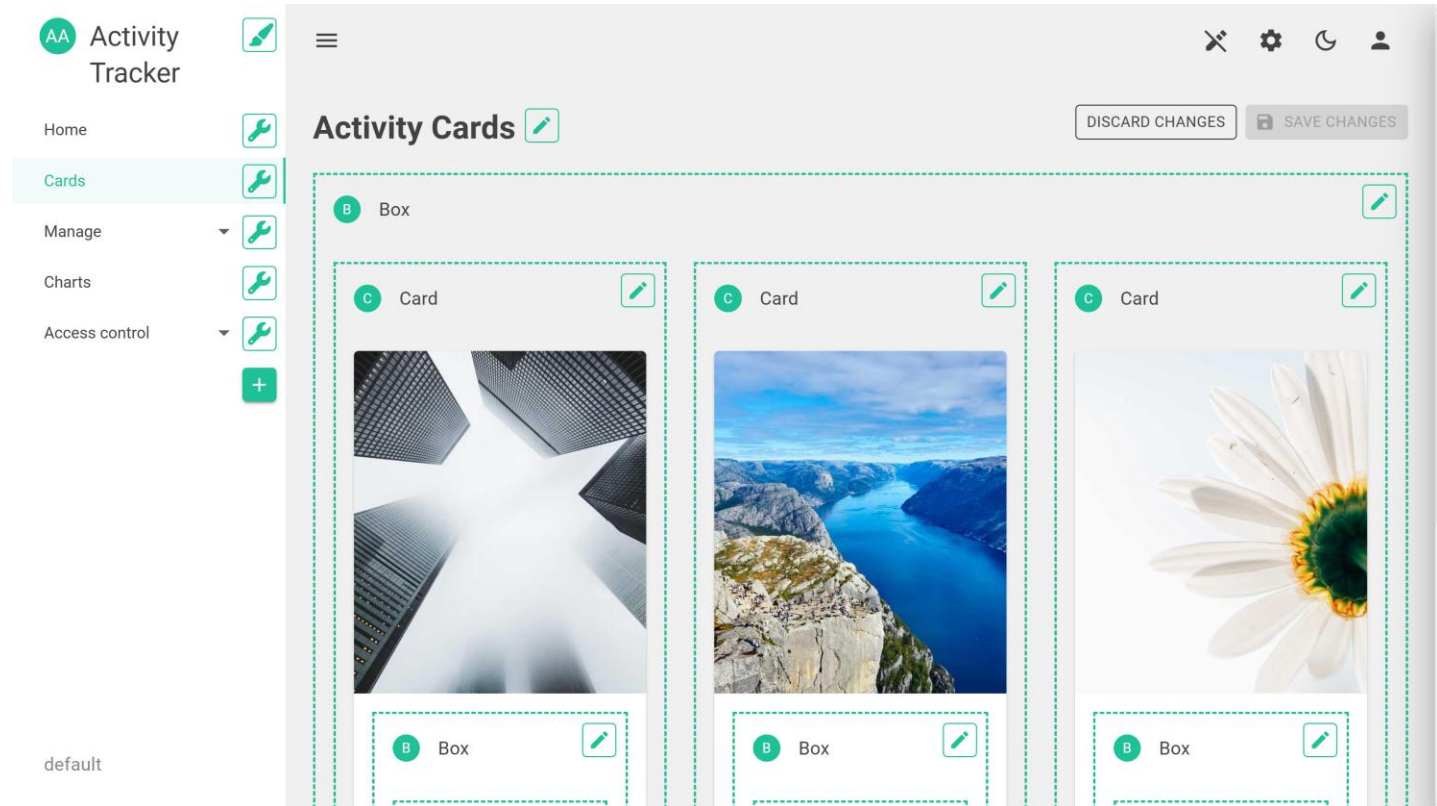
### In-App (Runtime) ⚡

Model live in the browser—fast, visual, and intuitive.

### In Visual Studio 🛠️

Model with full compile-time safety and code quality.

# In-App modeling



# In-App Logic Blocks

AA Activity Tracker

Home

Cards

Manage

Activities

Categories

Charts

Access control

default

## Logic block

Name\*  
AddActivityComment

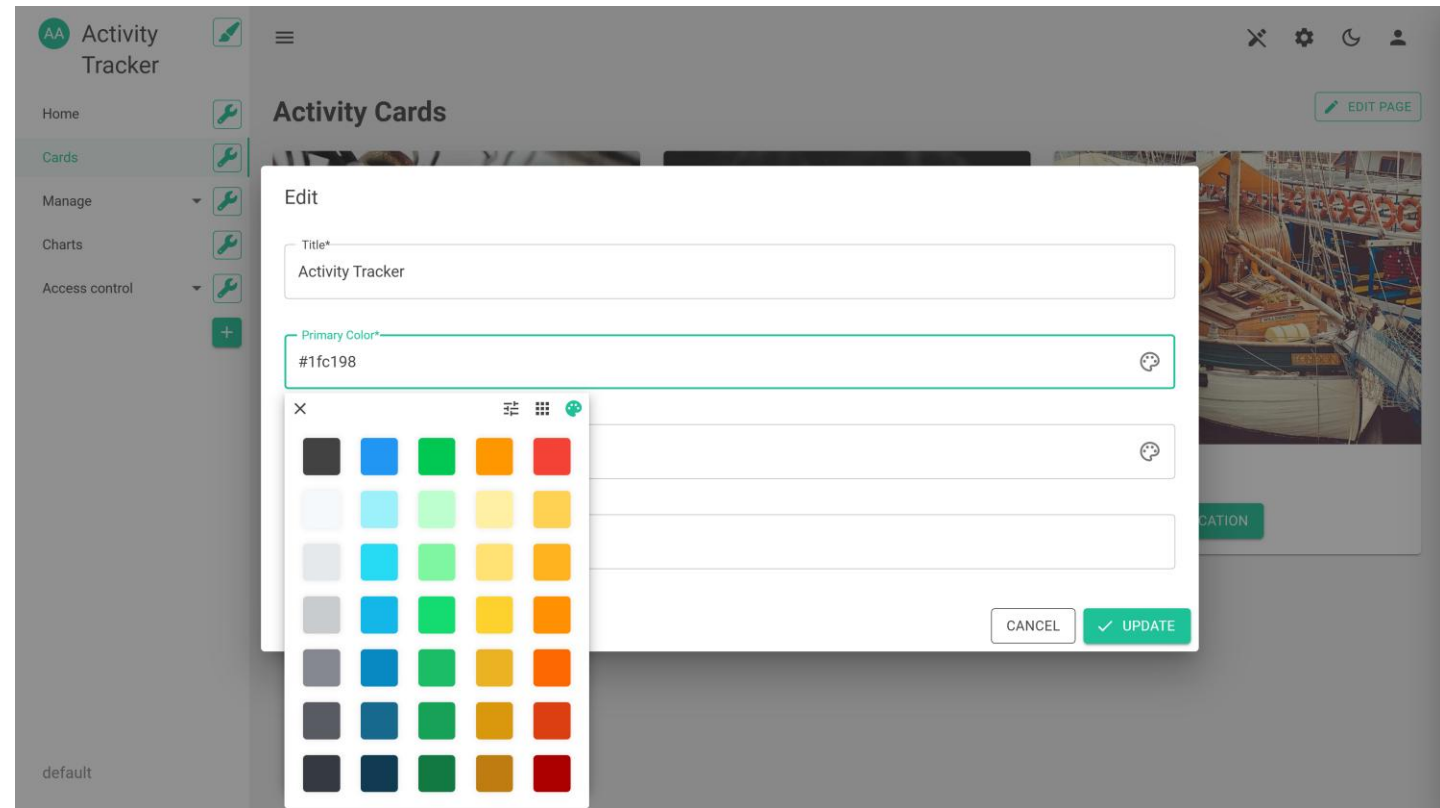
Kind\*  
Blockly

- Logic
- Loops
- Math
- Text
- Lists
- Variables
- Functions
- Messaging
- User Interface
- Common
- Data
- User

```
set userName to get user name
set existingComments to get property "Comments"
set property "Comments" with value
  create text with existingComments
  "This activity was created by: "
  userName
  "!"
```



Edit the app's  
style directly in  
the app



# Modeling using Visual Studio

```
var builder = new AppBuilder(AppConstants.App.Name, "Activity Tracker", "#1fc198", "AA");
builder
    .AddEntityStorage(globalStorage)
    .AddEntityStorage(defaultStorage)

    .AddEntityClass<Organization>(globalStorage)
    .AddSimpleCrudForClass<Organization>(accessControl)
    .AddBasicAccessControl<UserAccount, OrganizationTenant>(globalStorage, accessControl)

    .AddEntityClass<Activity>(defaultStorage)
    .AddEntityClass<Category>(defaultStorage)

    .AddLogicBlock(new LogicBlock() { ... })
    .AddLogicBlock(new LogicBlock() { ... })
    .AddLogicBlock(new LogicBlock() { ... })
    .AddPage(new Page()
    {
        Id = AppConstants.Pages.HomePage,
        Title = "Home",
        Name = "home",
        PageArea = new BoxComponent()
        {
            Items = [
                new BoxItem() {
                    Component = new HtmlComponent() {
                        HtmlContent = "Welcome to this activity app! 🐼 Stay organized, boost produ
                    },
                    Size = 12,
```



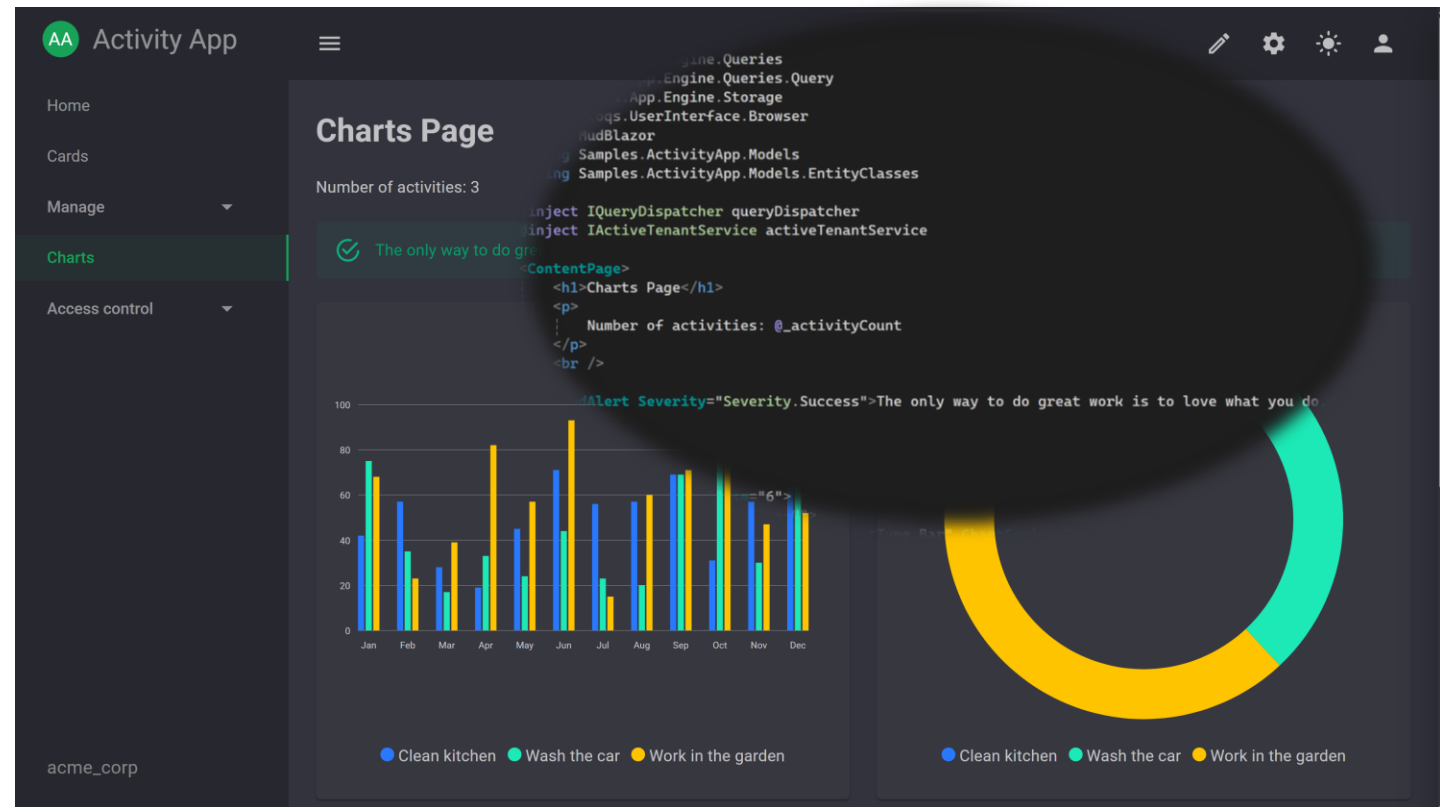
# Programming logic using Command & Query dispatcher pattern

- Feature-based architecture.
- Works with Blazor WASM & API.
- Leverages powerful Dependency Injection.

```
[CommandHandler<Activity>]
0 references | Tom van Oost, 3 hours ago | 1 author, 1 change
public class ActivityCreate : ICommandHandler<EntityCreateCommand, EntityCreateCommandResult>
{
    2 references | Tom van Oost, 3 hours ago | 1 author, 1 change
    public Task<EntityCreateCommandResult> HandleAsync(EntityCreateCommand command, CancellationToken cancellationToken)
    {
        var newActivity = new Activity()
        {
            Name = "New Activity",
            Description = "New Activity Description",
        };

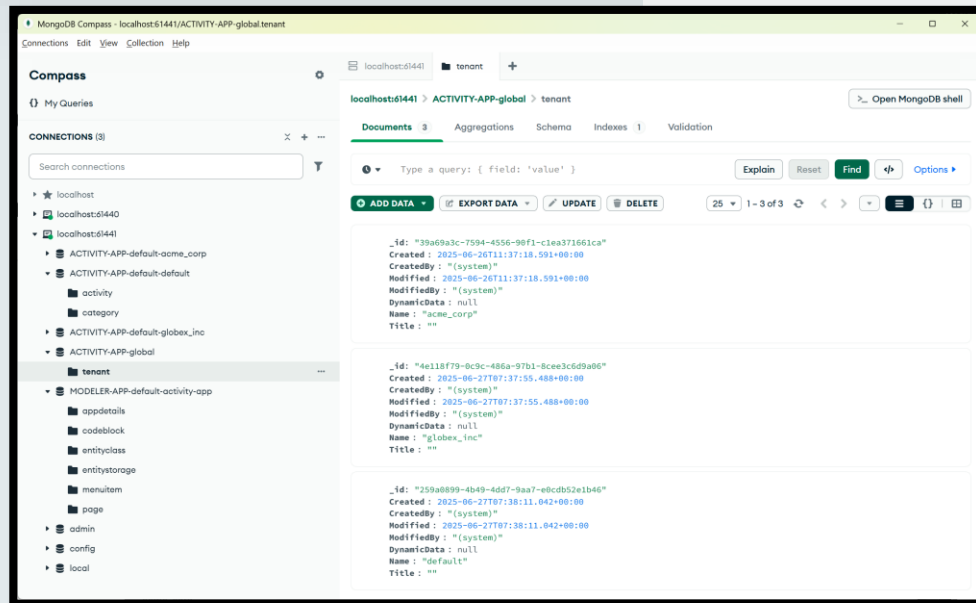
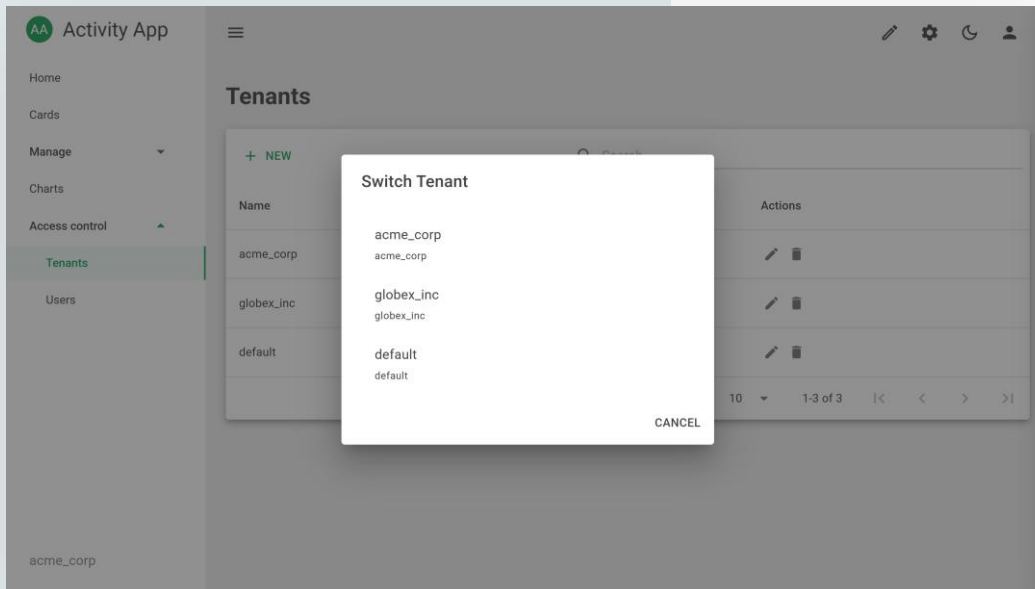
        return Task.FromResult(EntityCreateCommandResult.CreateSuccess(newActivity));
    }
}
```

Add custom  
components,  
pages and API  
endpoints with  
ease



# Access control

- Authentication using JWT & OIDC (OpenID Connect).
- Multi-tenancy support.
- Users, permissions, roles.
- Automatic database creation.





# Define multiple data stores on global or tenant level

AA Activity Tracker

Home

Cards

Manage ▲

Activities

Categories

Charts

Access control ▼

default

Entity storage

Name\*

global

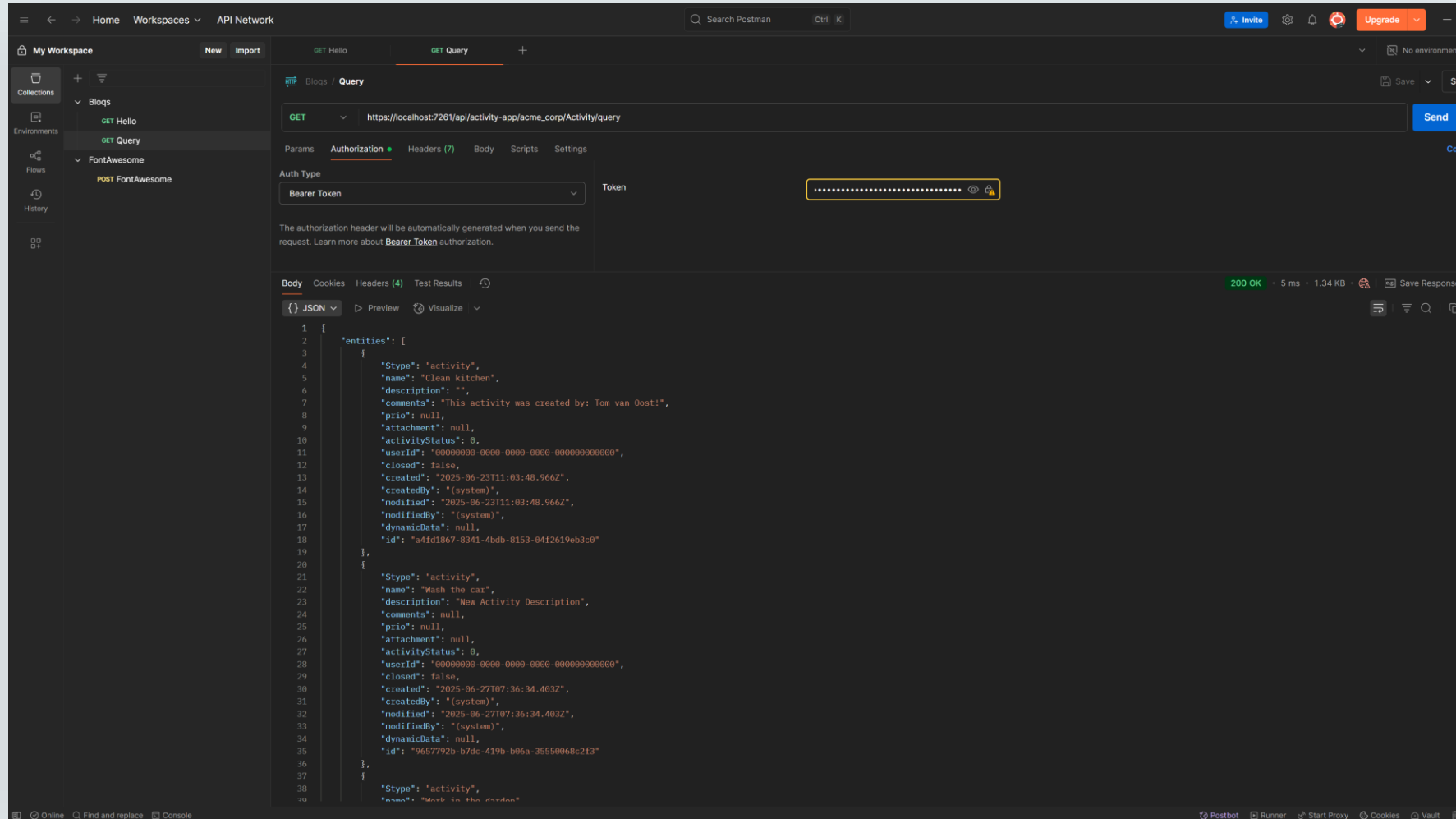
Entity Storage Type\*

Global ▲

Global

Tenant

# Expose a robust REST API, protected with JWT authentication



# Explore the application model via the REST API

FontAwesome  
POST FontAwesome

Key	Value	Description
Key	Value	Description

Body Cookies Headers (4) Test Results

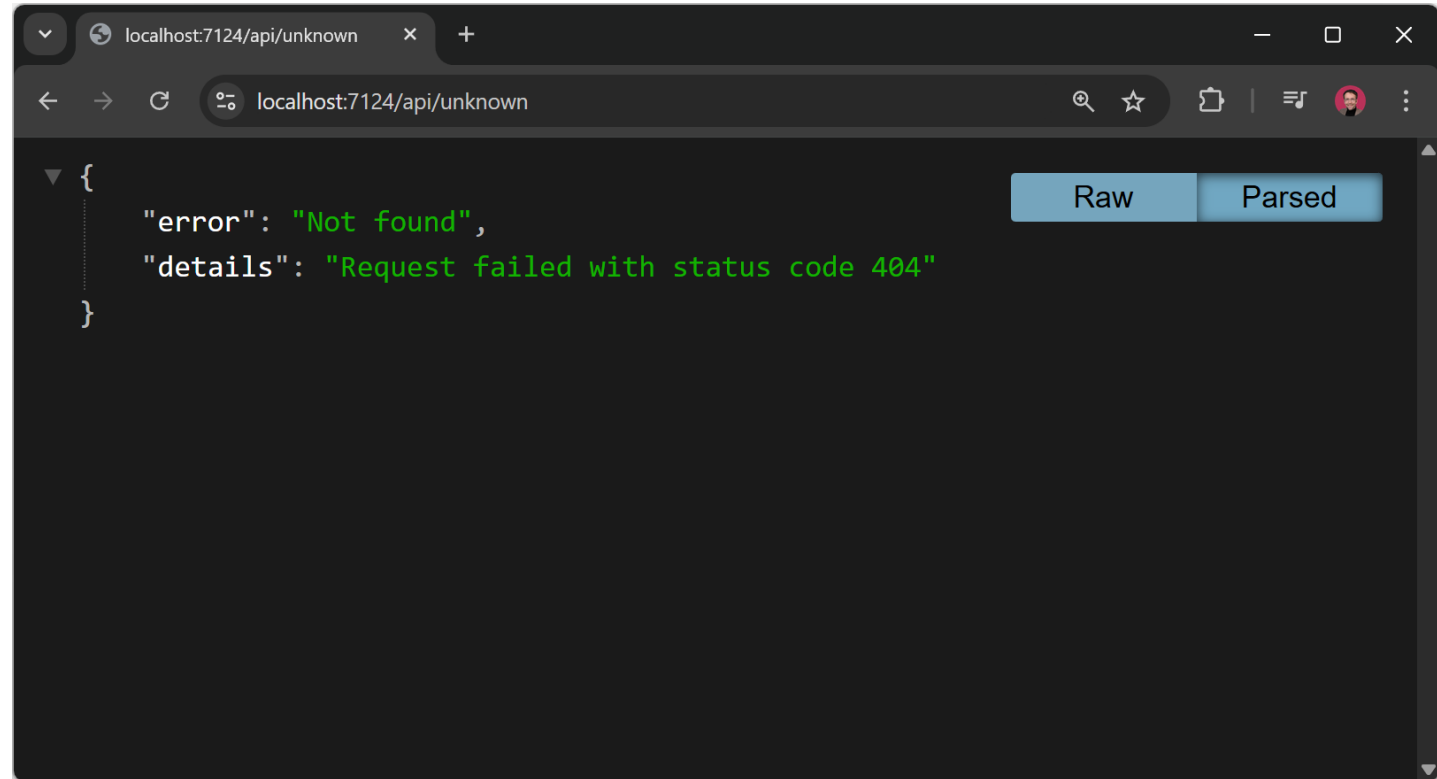
200 OK · 16 ms · 63.71 KB · Save Response

JSON Preview Visualize

```
217     "modifiedBy": "(system)",
218     "dynamicData": null,
219     "id": "6cd54dc9-b947-b586-107b-8115e38d7cb0"
220   },
221   {
222     "name": "Activity",
223     "plural": "Activities",
224     "properties": [
225       {
226         "name": "Name",
227         "dataType": 0,
228         "propertyEntityClassId": null,
229         "isRequired": true,
230         "isInvisible": false,
231         "validPropertyValues": null,
232         "created": "2025-06-26T11:37:01.846Z",
233         "createdBy": "(system)",
234         "modified": "2025-06-26T11:37:01.846Z",
235         "modifiedBy": "(system)",
236         "dynamicData": null,
237         "id": "6e36fb64-ac28-e243-589e-b49f55601ac1"
238       },
239       {
240         "name": "Description",
241         "dataType": 0,
242         "propertyEntityClassId": null,
243         "isRequired": false,
244         "isInvisible": false,
245         "validPropertyValues": null,
246         "created": "2025-06-26T11:37:01.846Z",
247         "createdBy": "(system)",
248         "modified": "2025-06-26T11:37:01.846Z",
249         "modifiedBy": "(system)",
250         "dynamicData": null,
251         "id": "ece29084-b1de-765a-aad9-f4fe955031eb"
252       },
253       {
254         "name": "Comments",
255         "dataType": 0,
256         "propertyEntityClassId": null,
257         "isRequired": false
```

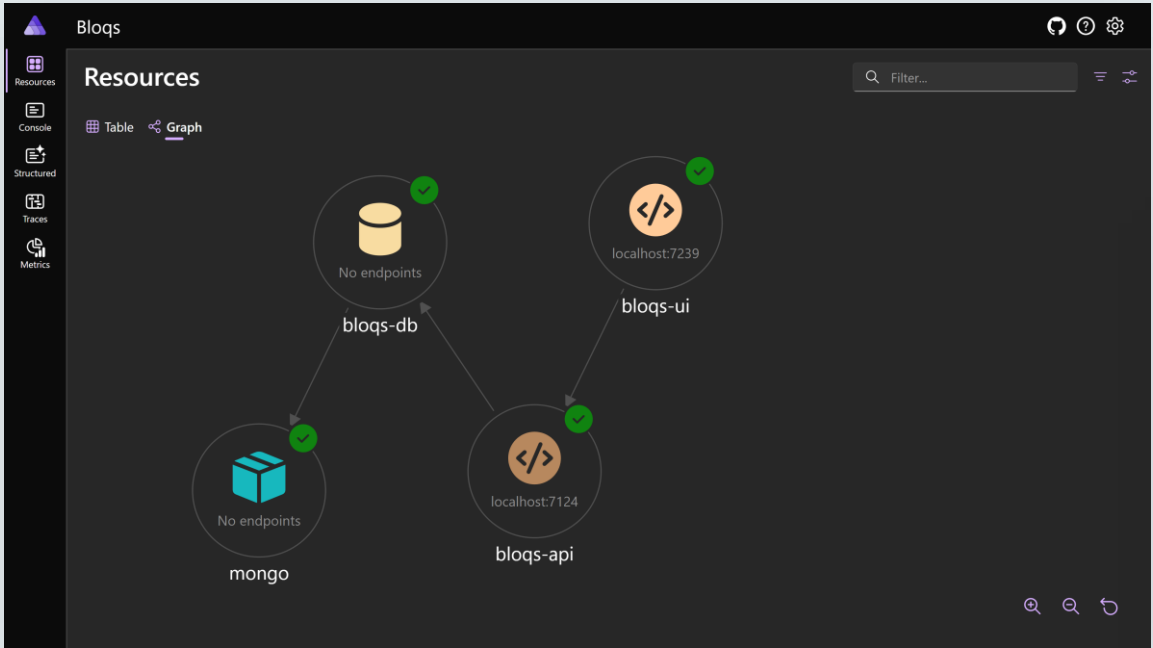


## Clear and Standardized API Error Handling



```
{  
  "error": "Not found",  
  "details": "Request failed with status code 404"  
}
```

# Deploy using .NET Aspire



The screenshot shows the 'Console logs' page in the .NET Aspire dashboard, specifically for the 'bloqs-api' resource. The interface includes a sidebar with navigation options: Resources, Console, Structured, Traces, and Metrics. The main area displays a list of log messages. The logs show the application waiting for other resources to start, then successfully starting and listening on the specified ports.

```
1 Waiting for resource 'mongo' to enter the 'Running' state.
2 Waiting for resource 'bloqs-db' to enter the 'Running' state.
3 Waiting for resource 'mongo' to become healthy.
4 Waiting for resource ready to execute for 'bloqs-db'.
5 Finished waiting for resource 'bloqs-db'.
6 Waiting for resource ready to execute for 'mongo'.
7 Finished waiting for resource 'mongo'.
8 info: Microsoft.Hosting.Lifetime[14]
9   Now listening on: https://localhost:61895
10 info: Microsoft.Hosting.Lifetime[14]
11   Now listening on: http://localhost:61896
12 info: Microsoft.Hosting.Lifetime[0]
13   Application started. Press Ctrl+C to shut down.
14 info: Microsoft.Hosting.Lifetime[0]
15   Hosting environment: Development
16 info: Microsoft.Hosting.Lifetime[0]
17   Content root path: C:\Estiom\git\Bloqs\Bloqs\Bloqs.Api
```

