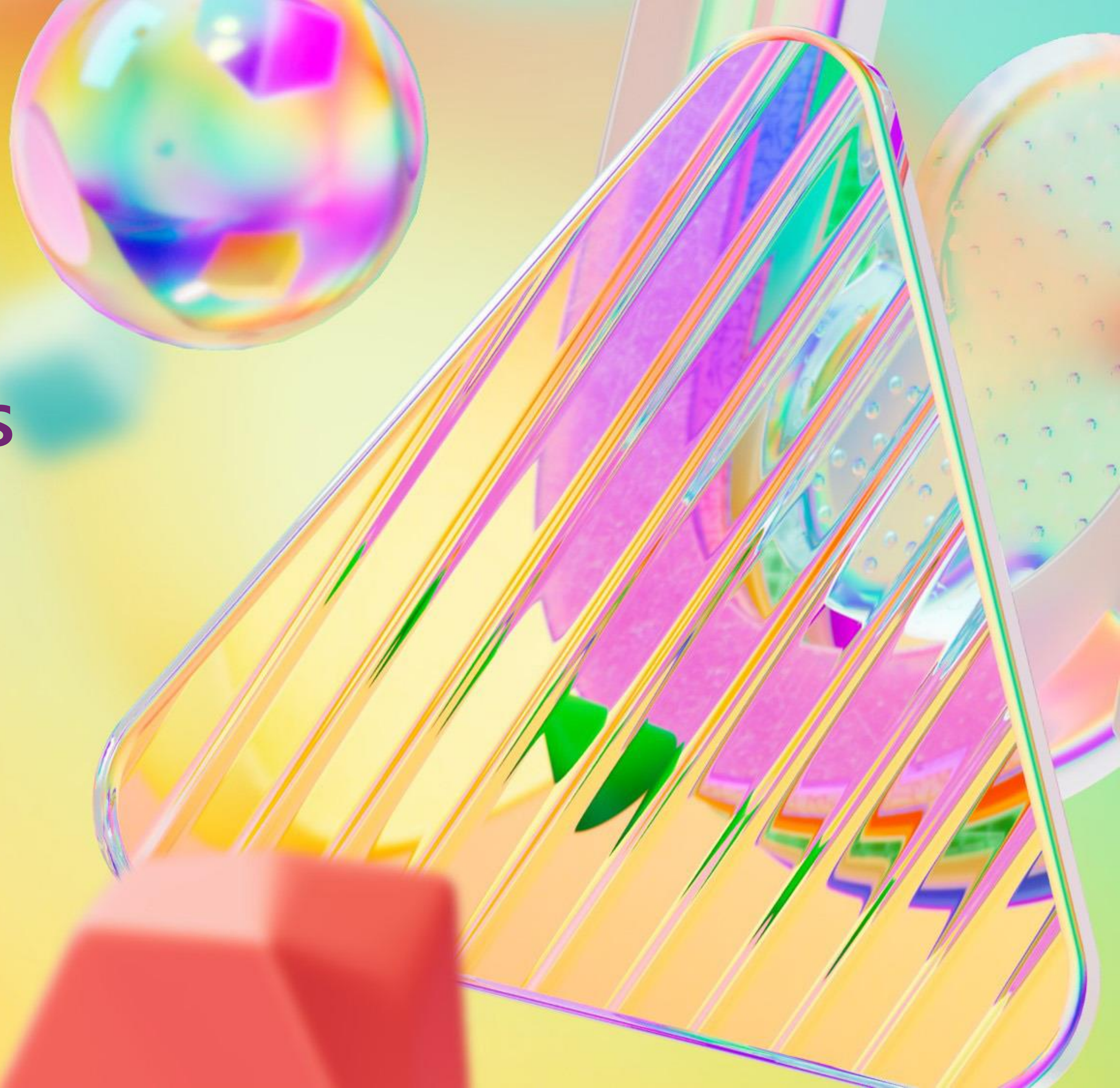
Microsoft

Modernize old web applications to .NET 8

Tomas Herceg



Tomas Herceg

Microsoft MVP
CEO @ RIGANTI
@hercegtomas

Ask me about

.NET & web development

Azure

Maintaining an open-source framework

Writing book about modernization

Why modernize?

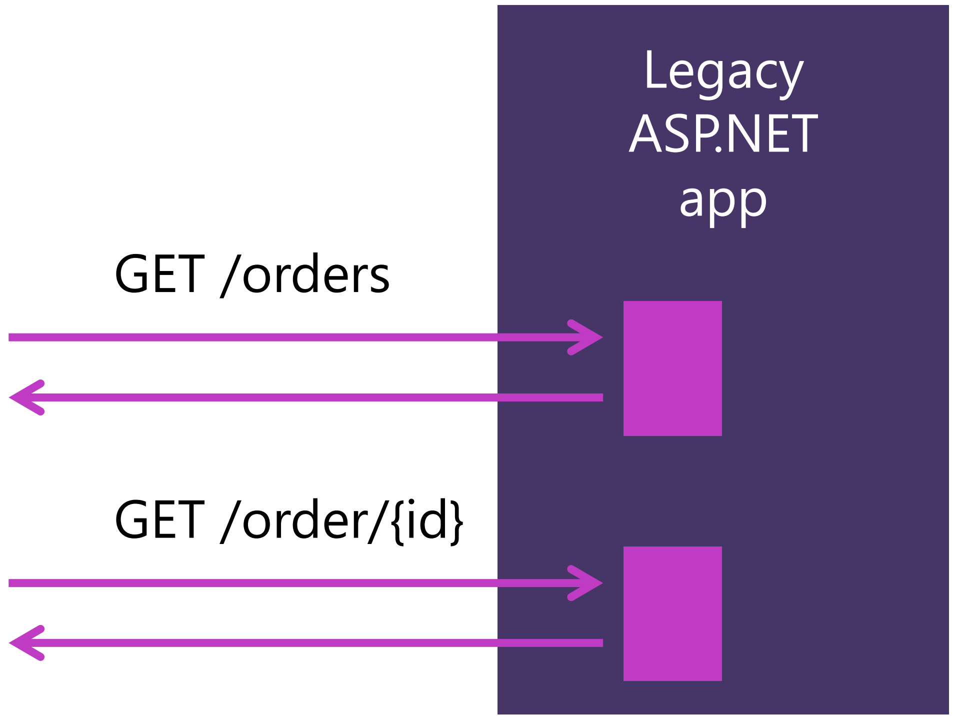
- Access all the benefits of the new .NET
  - New C# language features
  - Better performance
  - New libraries – logging, dependency injection, configuration…
  - Support for Linux & containers

- Easier to hire new developers
  - Nobody wants to work with old projects

Method 1: Side-by-side modernization

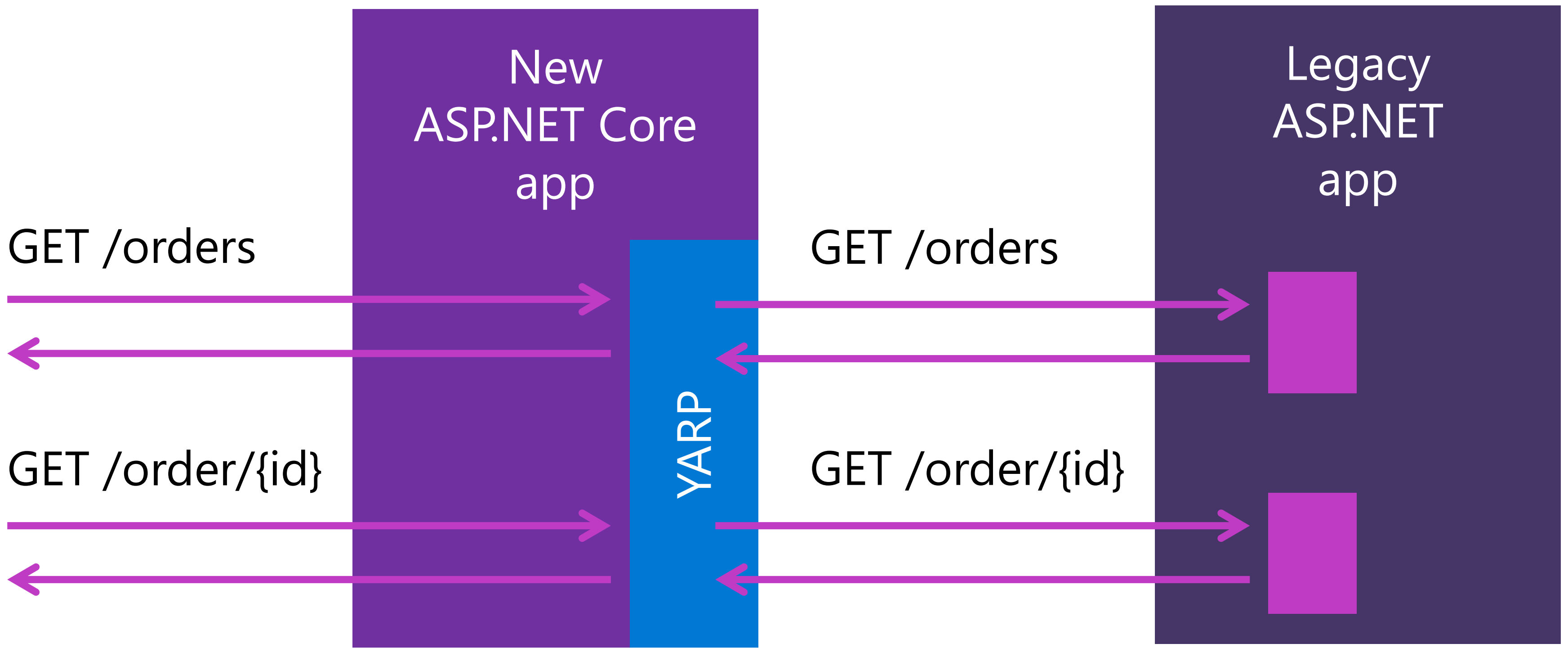
Side-by-side modernization

- Create an empty ASP.NET Core application
- Use YARP to send unhandled requests to the old one



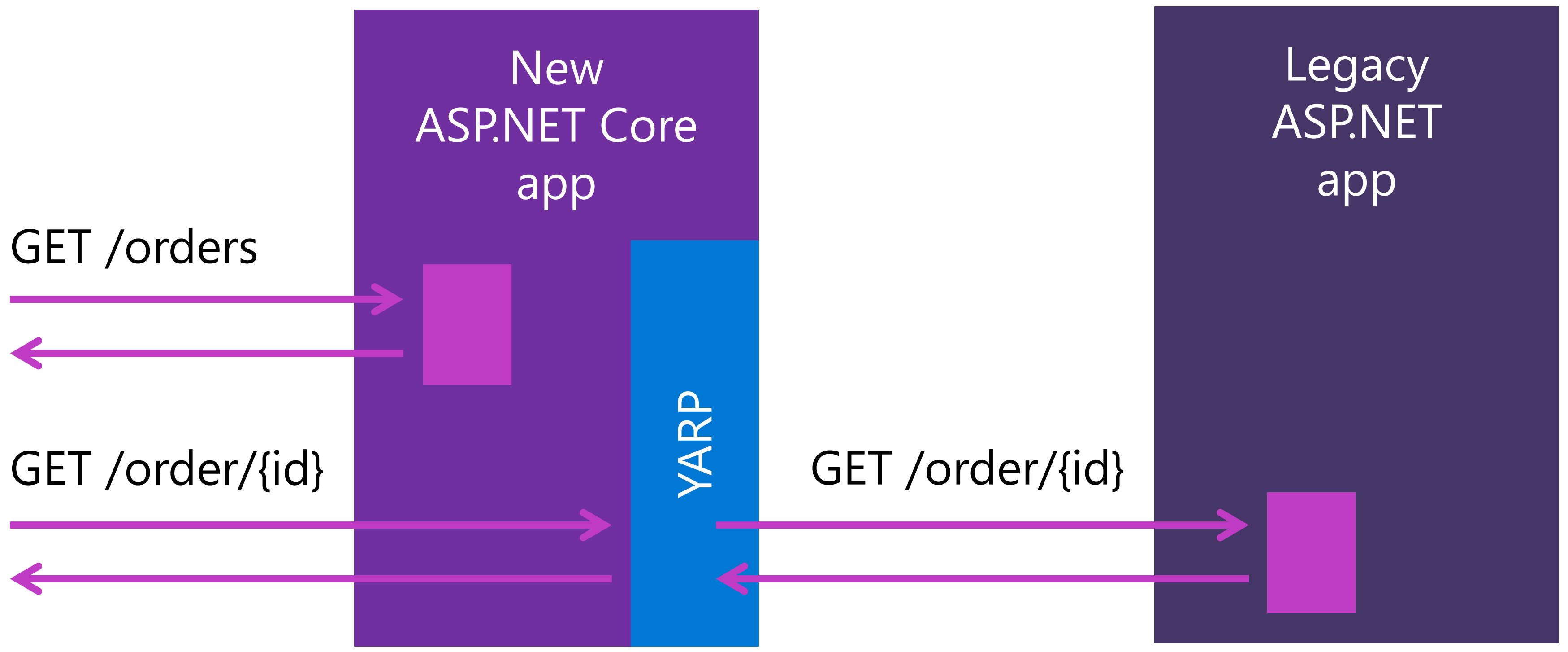
Side-by-side modernization

- Create an empty ASP.NET Core application
- Use YARP to send unhandled requests to the old one



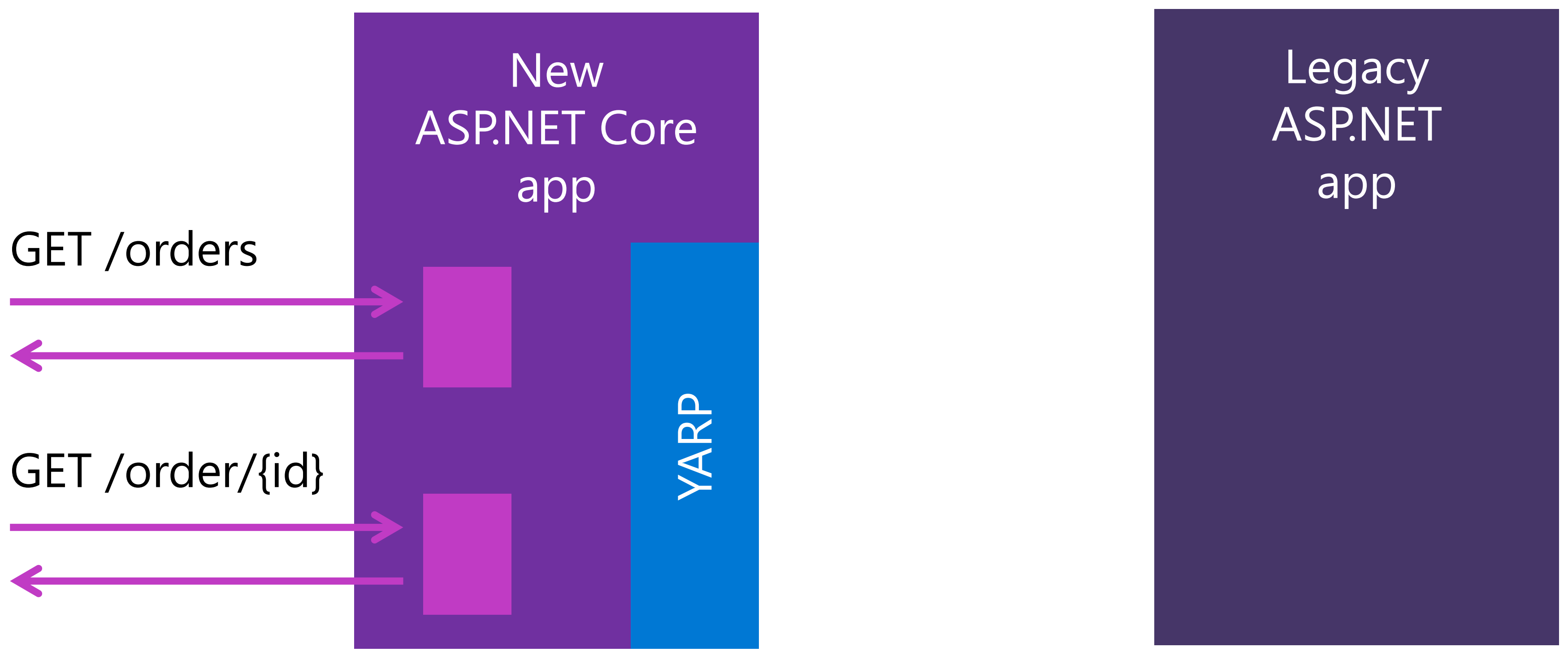
Side-by-side modernization

- Create an empty ASP.NET Core application
- Use YARP to send unhandled requests to the old one



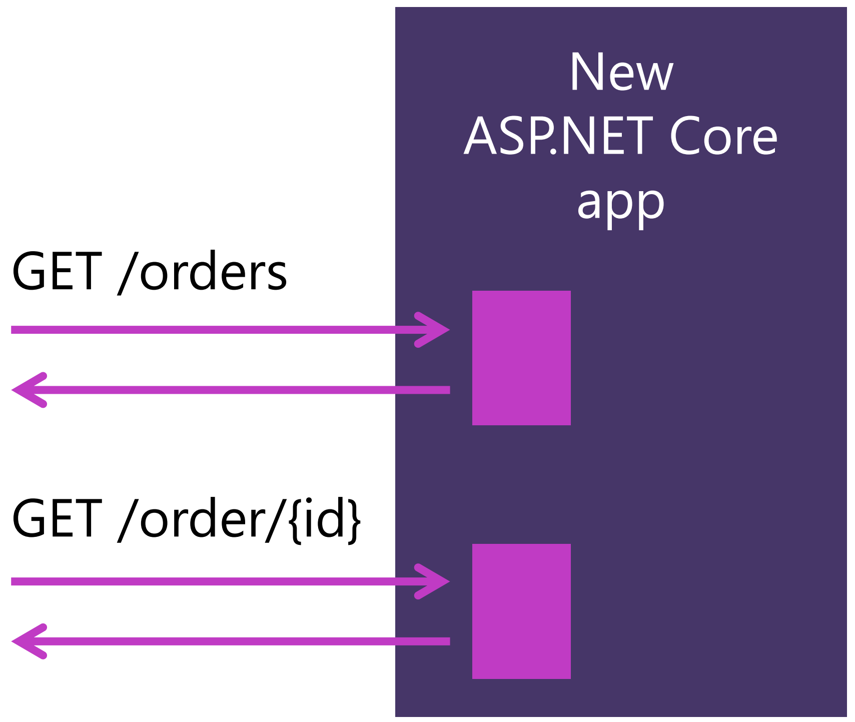
Side-by-side modernization

- Create an empty ASP.NET Core application
- Use YARP to send unhandled requests to the old one



Side-by-side modernization

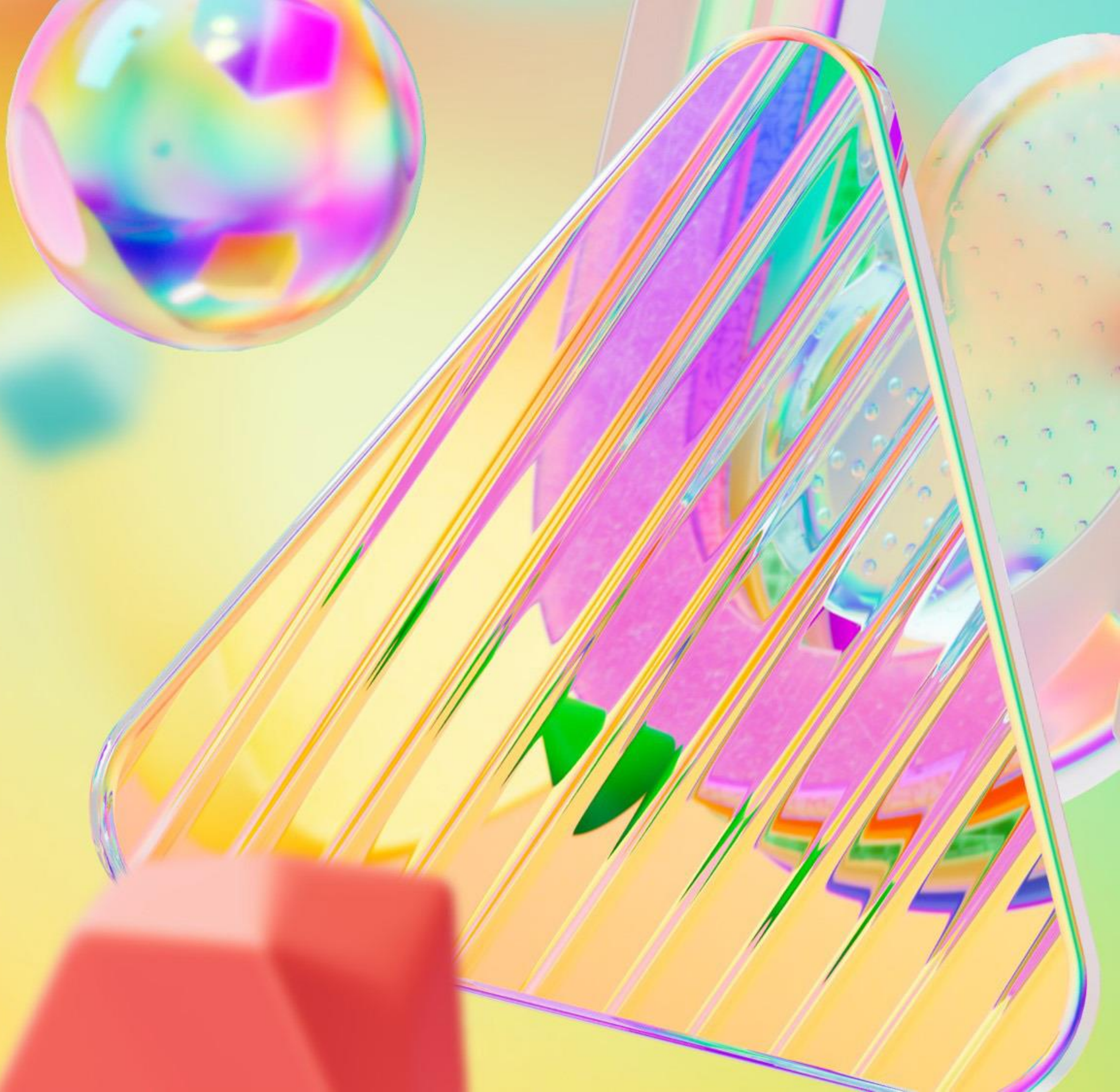
- Create an empty ASP.NET Core application
- Use YARP to send unhandled requests to the old one



Microsoft

DEMO

Side-by-side modernization



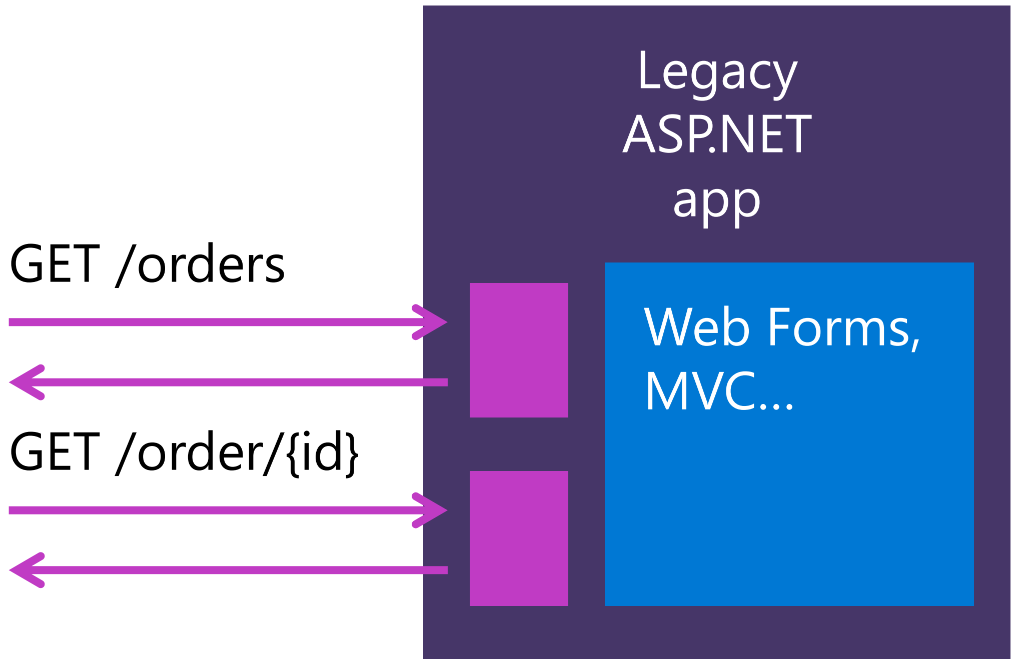
Side-by-side modernization challenges

- Authentication
  - Different format of auth cookies in ASP.NET and ASP.NET Core
  - Use an external identity provider or implement a custom SSO
- Session
  - Store it in the database in a format understandable to both applications
- Caching
  - Invalidate the cache in the other application, or introduce distributed caching
- Concurrency and shared resources
  - Filesystem, in-process locking…

Method 2: In-place modernization

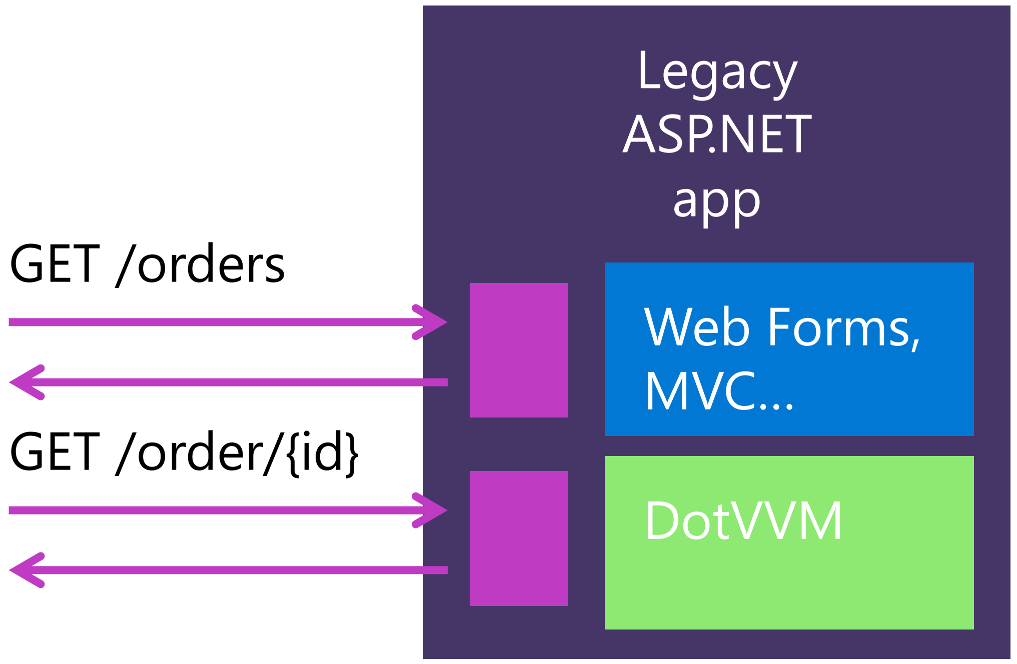
In-place modernization

- Use a UI framework that supports both .NET Framework and .NET 8
  - DotVVM – open-source, member of the .NET Foundation, MVVM approach
- Install it in the legacy app and rewrite the pages in-place
- Afterwards, switch the project to .NET 8



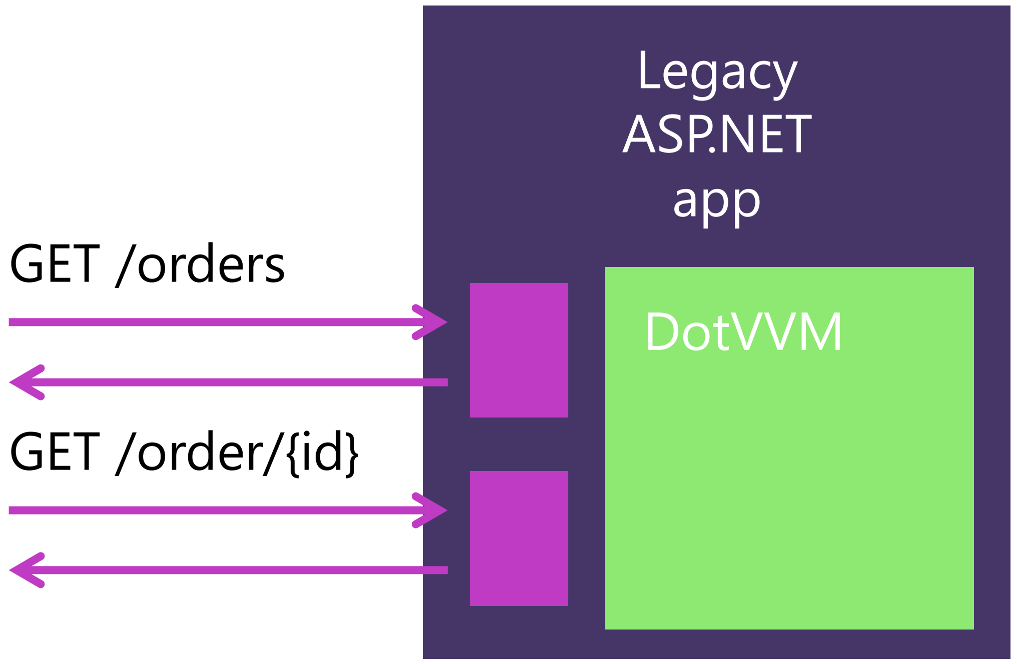
In-place modernization

- Use a UI framework that supports both .NET Framework and .NET 8
  - DotVVM – open-source, member of the .NET Foundation, MVVM approach
- Install it in the legacy app and rewrite the pages in-place
- Afterwards, switch the project to .NET 8



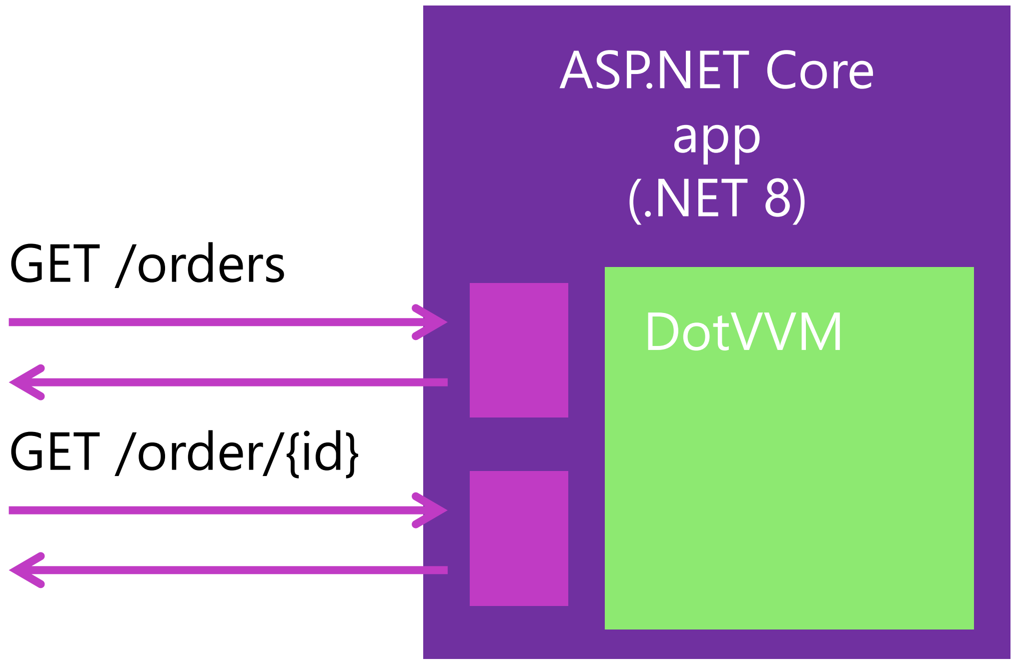
In-place modernization

- Use a UI framework that supports both .NET Framework and .NET 8
  - DotVVM – open-source, member of the .NET Foundation, MVVM approach
- Install it in the legacy app and rewrite the pages in-place
- Afterwards, switch the project to .NET 8



In-place modernization

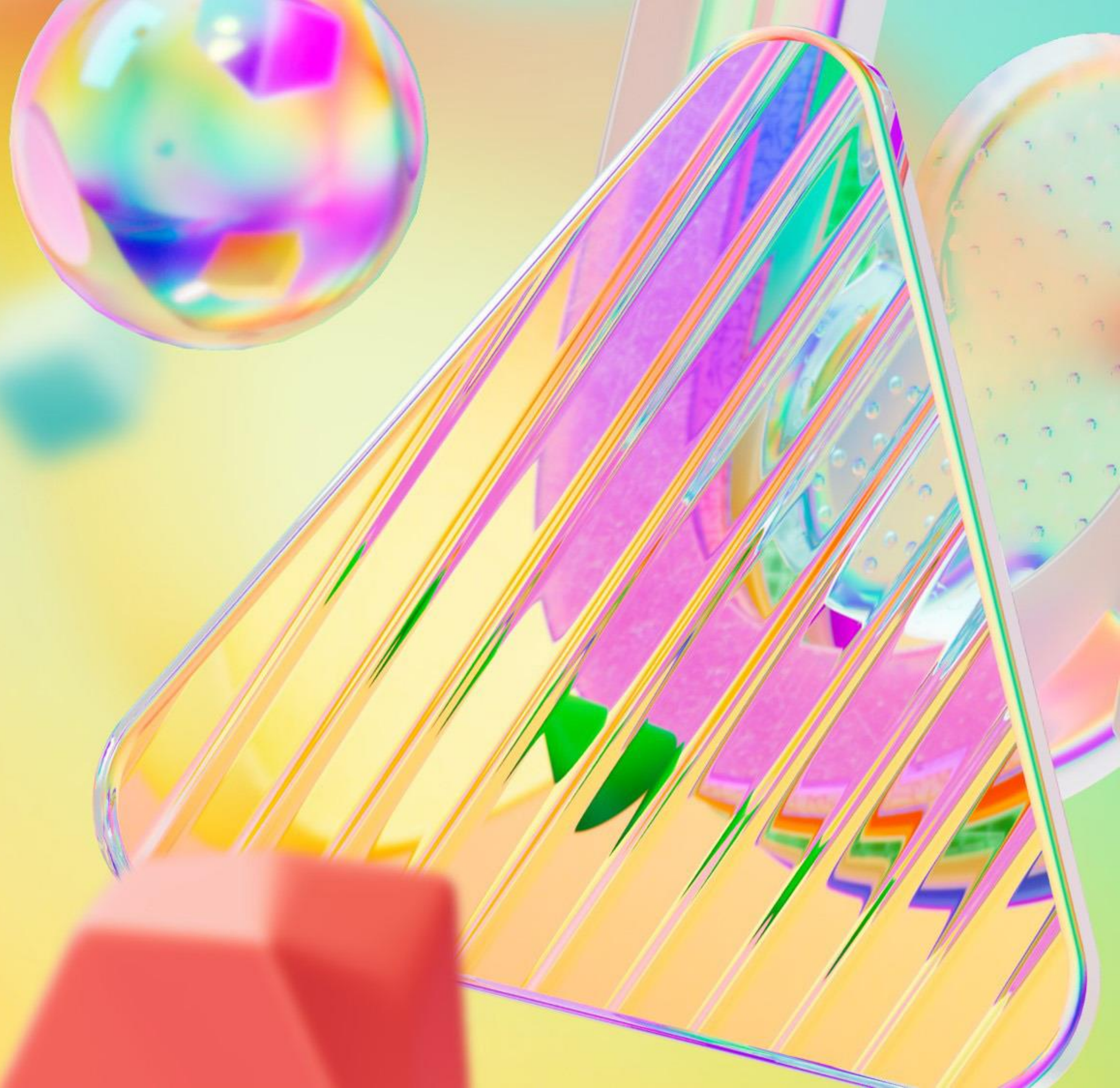
- Use a UI framework that supports both .NET Framework and .NET 8
  - DotVVM – open-source, member of the .NET Foundation, MVVM approach
- Install it in the legacy app and rewrite the pages in-place
- Afterwards, switch the project to .NET 8



Microsoft

DEMO

In-place modernization



In-place modernization challenges

- You must use a framework supporting both old and new .NET
  - There are not many options
  - DotVVM is the only full-featured framework I am aware of

Choose the method that suits your scenario.

Thank you!

- We appreciate your time with us
- For more info: tomas.herceg@riganti.cz
- Stop by the Expert Meet-up to get your questions answered
- Your feedback is important! Visit https://aka.ms/MicrosoftBuildEvals to complete your session evaluations

Microsoft