

Cloud Native meets AI

Moderne Cloud Native Backends bauen mit
.NET Aspire und Microsoft.Extensions.AI

Heute auf der Karte

- Der Use Case: AI Information Extraction aus Text, Speech und Bildern
- .NET Aspire als Host für verteiltes Backend zur Entwicklungszeit
- Microsoft.Extensions.AI für Zugriff OpenAI API
- Endpunkt-Design und Prompt Management des AI-Services
- Ein kleiner Ausflug in Authentifizierung und Sicherheit

Zwischenfragen und Anmerkungen erlaubt!



<https://github.com/feO2x/cloud-native-meets-ai>

Kenny Pflug

[@feO2x](#)

Lead Software Engineer
TELIS/GWVS


- Seit 2009 professionell in .NET unterwegs
- Verteilte Systeme basierend auf ASP.NET (Core) seit 2014
- CLR und Framework Internals (Memory Management, Asynchronous Programming, Threading, ORMs, DI Containers, Serializers, etc.)



Use Case: ein komplexes Formular ausfüllen

Schadenmeldeformular

ADAC Versicherung AG · 81362 München



▼ Mitgliedsnummer

Schadenfall

in Deutschland ☐ im Ausland ☐

▼ Name / Vorname des Mitglieds

▼ PLZ / Ort

▼ Telefon tagsüber

▼ Mail / Fax

▼ Straße / Nr.

Angaben zum Schaden:

▼ Datum des Schadens

▼ Schadenland

▼ Aktueller Standort des Fahrzeugs

Wurde der ADAC bereits verständigt? ja ☐ nein ☐ Wenn nein

▼ Datum des Anrufs

▼ Uhrzeit

▼ PLZ / Ort des Schadens

▼ Zielort der Reise

▼ Grund:

▼ Uhrzeit

Panne ☐ Unfall ☐ Kfz-Diebstahl ☐ Sonst. ☐

▼ ca. Entfernung zum Wohnort

▼ Reisebeginn

Verständigung durch Polizei / Autobahnnotruf ja ☐ nein ☐

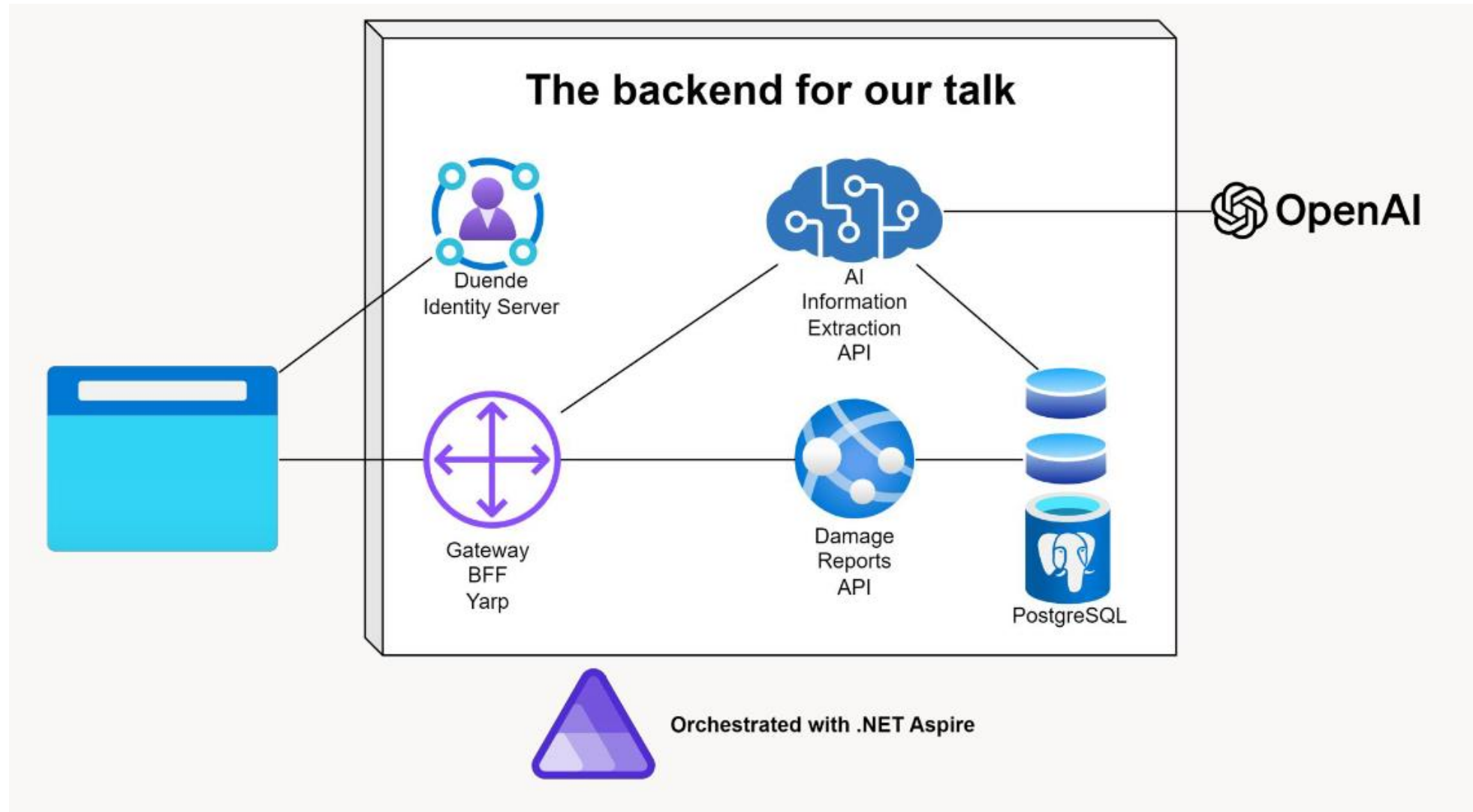
▼ Anz. d. Reisenden inkl. Fahrer

Können Sie die Mehrwertsteuer beim Finanzamt absetzen? ja ☐ nein ☐

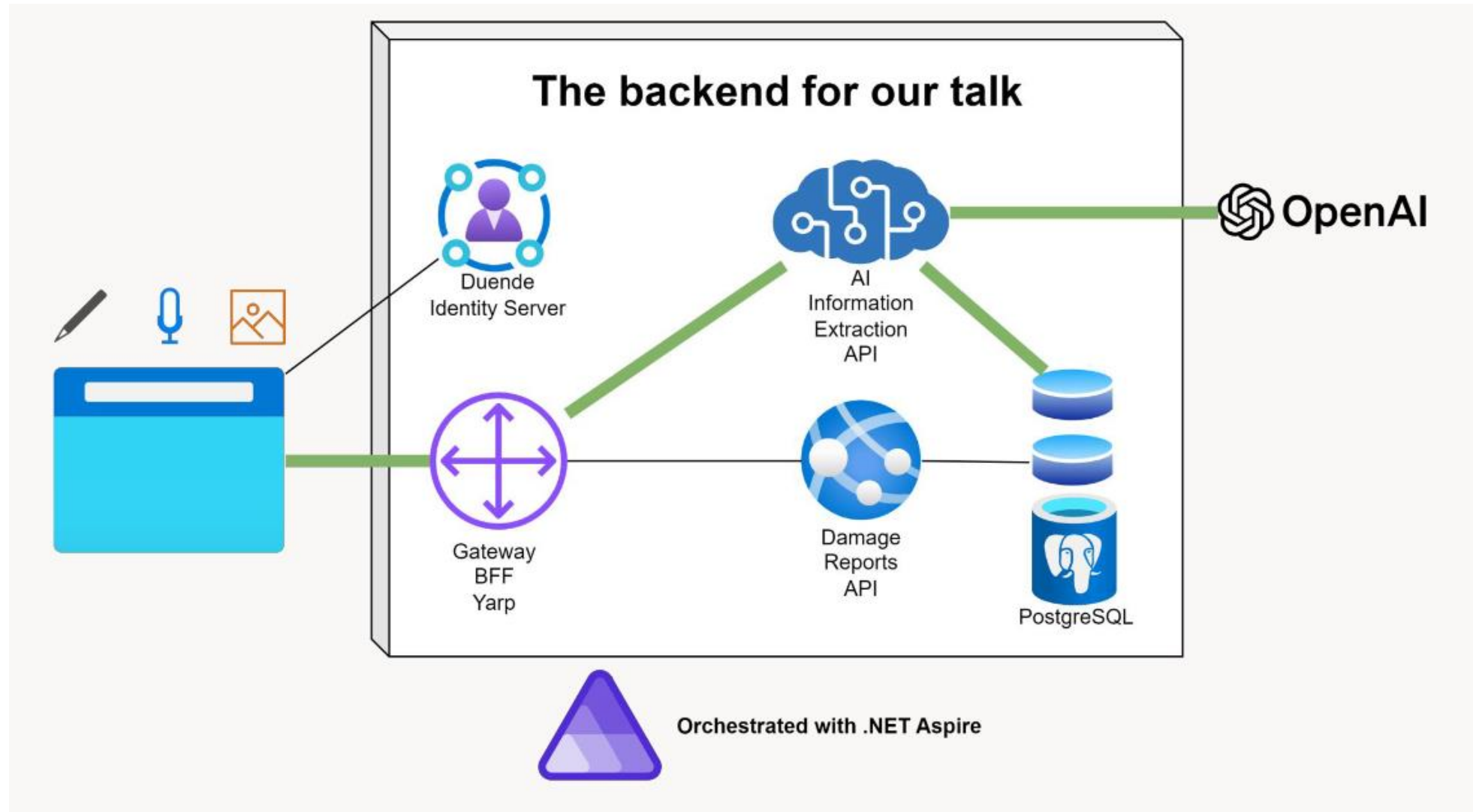
▼ geplante Rückreise

Demo Time!

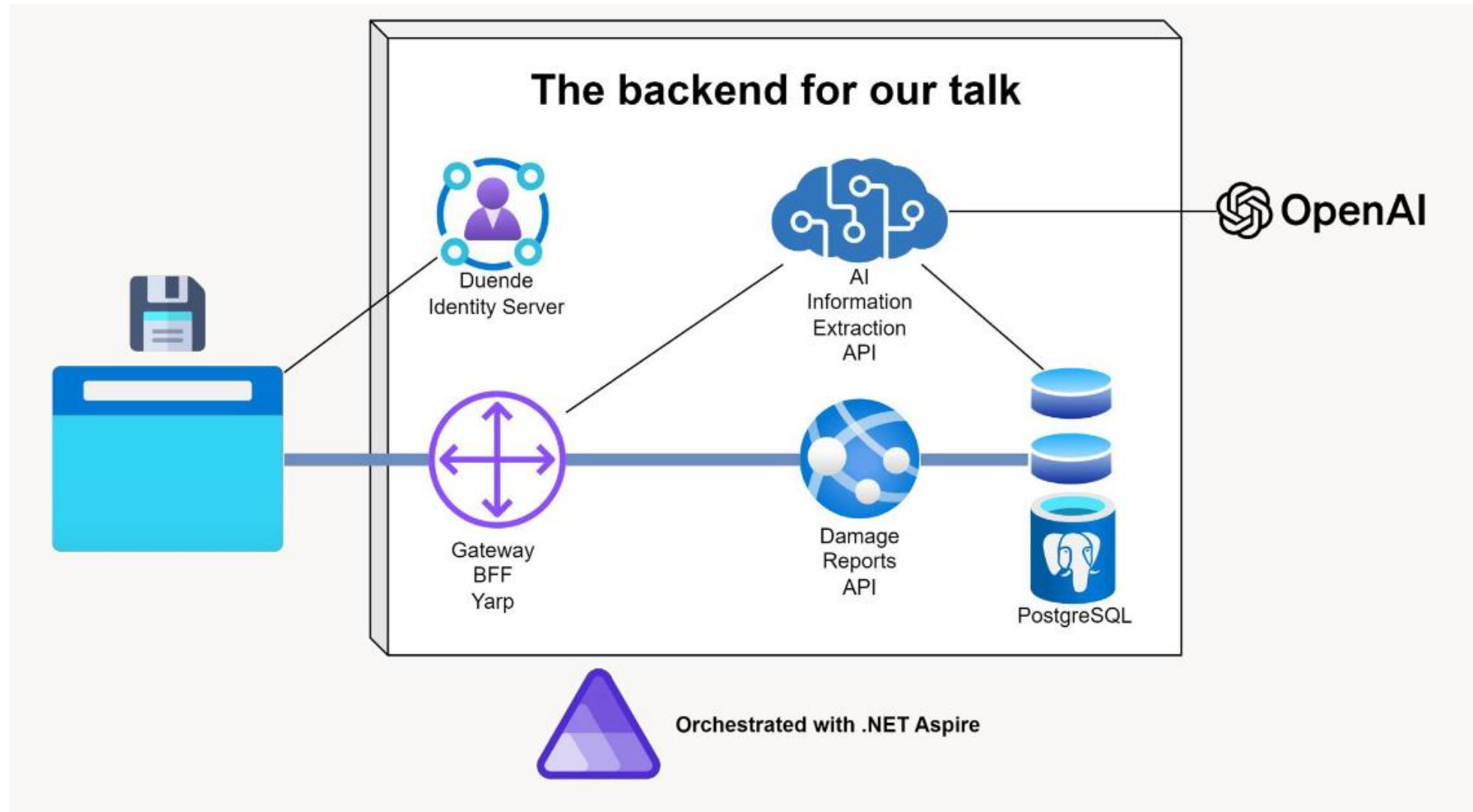
Übersicht über das Backend



Wenn das Formular ausgefüllt wird mit AI-Unterstützung



Beim Überprüfen und Speichern



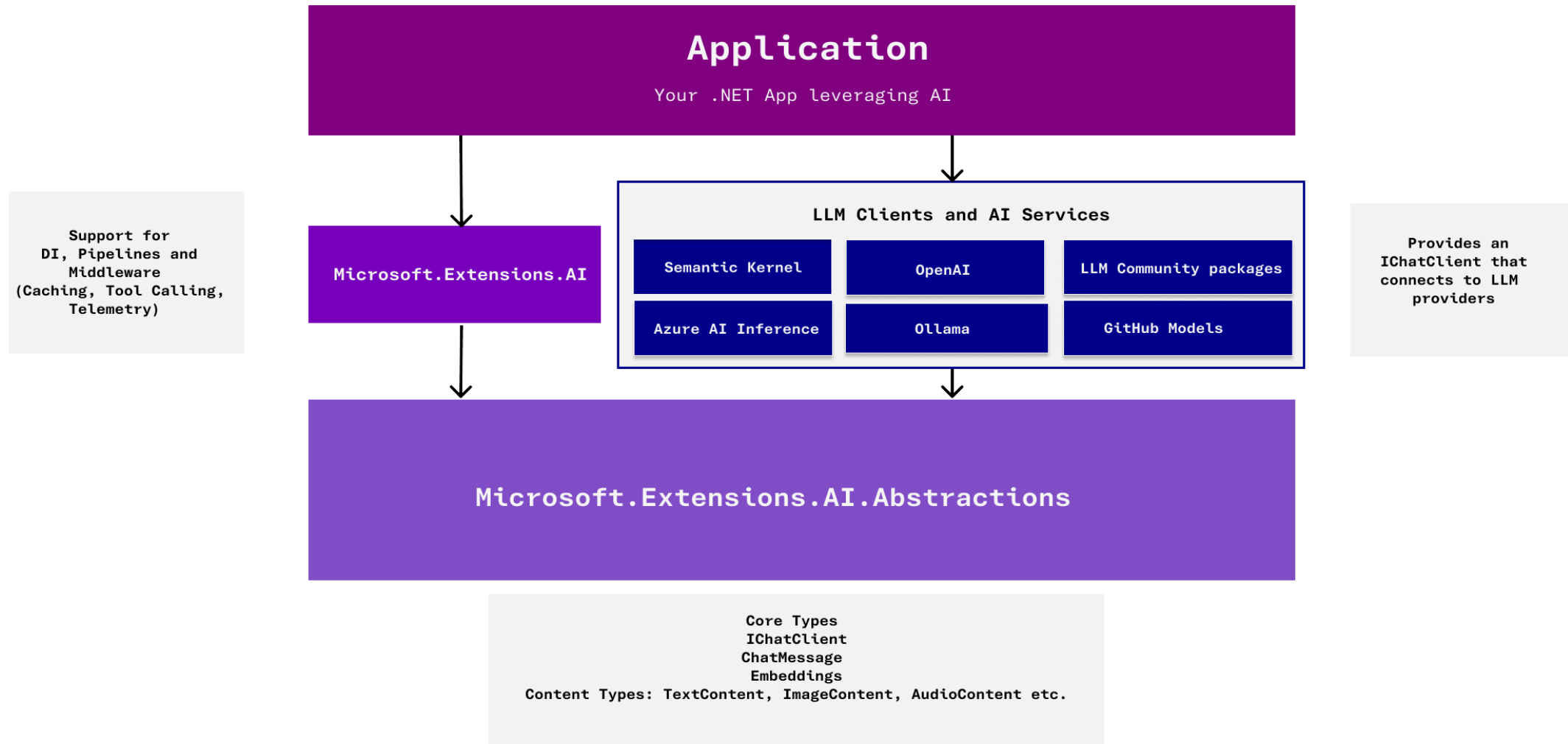
Cloud Native – die großen Themen



- Ein **Cloud Native Orchestrierungs-Toolkit**, geschrieben in .NET, das helfen soll, verteilte Systeme **bei der Entwicklung** einfacher zu **bauen**, zu **betreiben** und zu **beobachten**.
- Im Zentrum steht das **AppHost-Projekt**: dort definierst man Container und Projekte, welche von .NET Aspire orchestriert werden sollen.
- Aspire bietet integrierte Werkzeuge, Bibliotheken und Vorlagen (NuGet-Pakete, Projektvorlagen, CLI / IDE Integration) für Komponenten wie Datenbanken, Messaging, Caching, Telemetrie etc. – insbesondere **Client Integration** (in Service Projekte) und **Hosting Integration** (in App Host).
- Kein Fokus auf Produktions-Orchestrierung (wie z. B. Kubernetes), sondern **lokale Orchestrierung**, um Services **lokal starten, miteinander verbinden und debuggen** zu können – auch in **automatisierten Tests**.
- Via Manifest-Generierung, Integrationstemplates und abstrahierte Deploy-Pipelines kann Aspire als **Grundlage für Deployments** genutzt werden.
- Aspire legt großen Wert auf Observability / Telemetrie, Health Checks & Resilienz – festgelegt in einem **Shared Project** und angezeigt im **Aspire Dashboard**.

Demo Time!

Wie schreibt man einen Data Access Layer für AI?

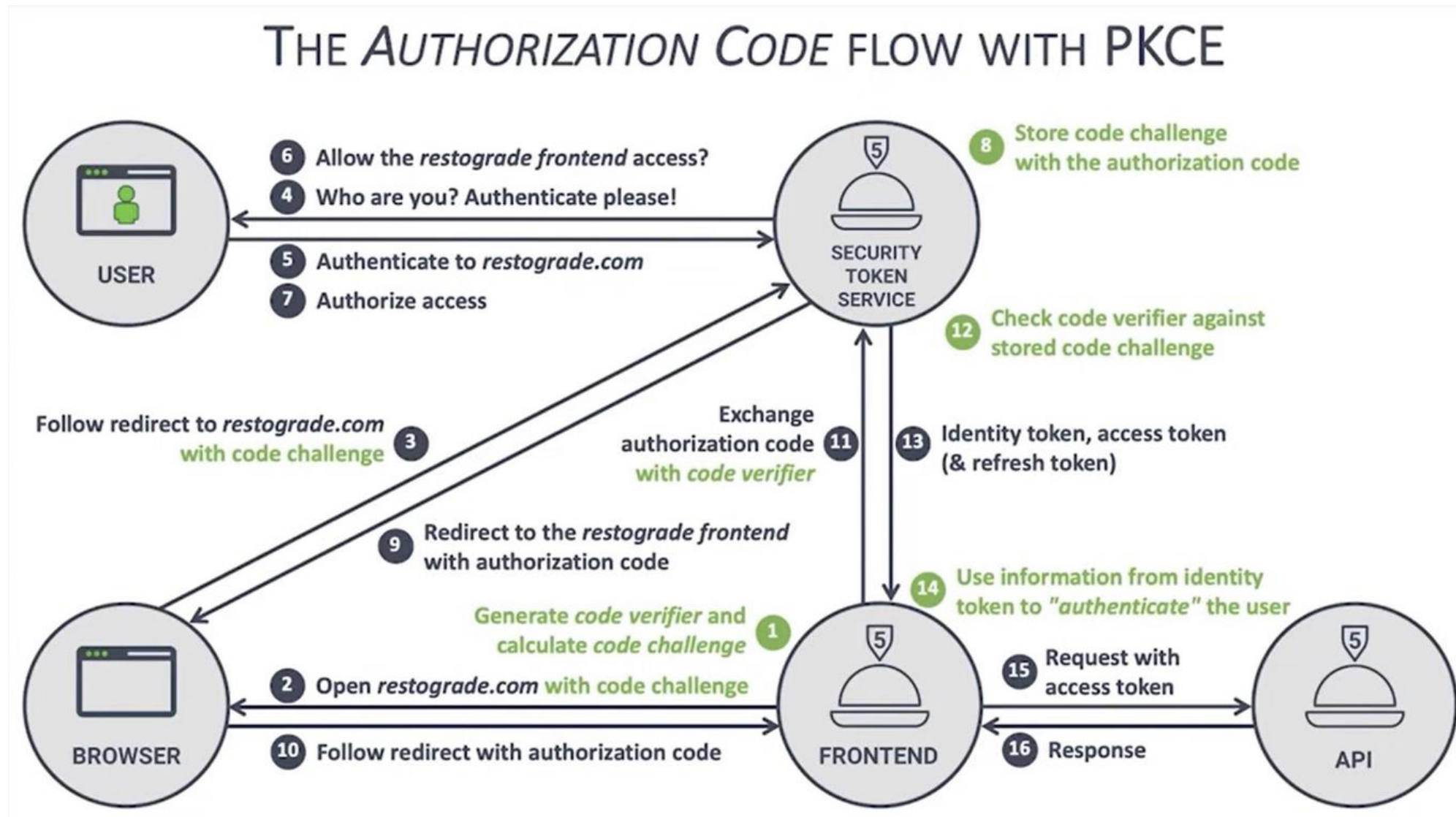


Demo Time!

Bitte keine JSON Web Tokens in den Browser bringen

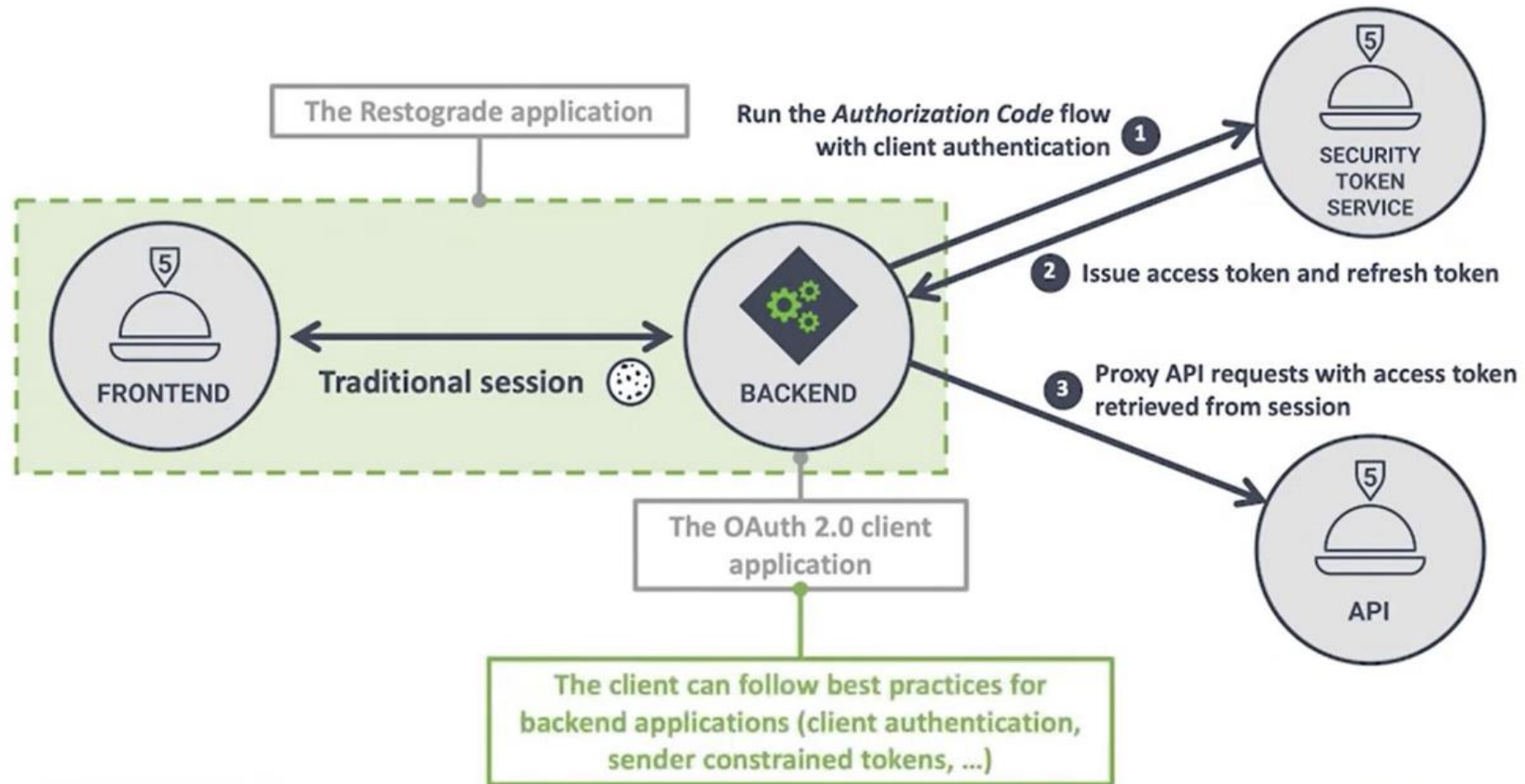
- Seit 5 bis 6 Jahren ist bekannt, dass **JSON Web Tokens (JWTs) nicht sicher** in **im Browser** laufenden Frontends aufbewahrt werden können.
- Stattdessen wird geraten, ein Backend for Frontend (**BFF**) einzusetzen, welches über einen angepassten OpenID Connect (OIDC) Authorization Code Flow **JSON Web Tokens auf der Backendseite verwaltet**.
- Zwischen Frontend und BFF wird dann **traditionelle Cookie Authentication** mit einem **http-only, signierten Cookie** eingesetzt. Zusätzliche Strategien wie **Content Security Policies (CSP)** sollten angewandt werden.
- Im BFF wird dann für einen eingehenden Request bestimmt werden, welcher User zu einem Cookie gehört. **Das Cookie wird** im Anschluss **durch das Access Token** für die Weiterleitung an API-Services **ersetzt**, automatische Refreshes sind möglich. Daher ergibt es Sinn, dass ein BFF gleichzeitig ein Reverse Proxy ist.
- Hinweis: AXA France hat einen anderen Ansatz entwickelt, der mit Demonstrated Proof of Possession (DPoP), CSPs und Service Workers einen neuen Ansatz entwickelt, wie man im JWTs im Browser sicher ablegen kann: <https://github.com/AxaFrance/oidc-client>
Ich kann diesen Ansatz nicht beurteilen und rate deshalb zum BFF!

OpenID Connect (OIDC) - Authorization Code Flow



OpenID Connect (OIDC) - Backend for Frontend (BFF) verwaltet Tokens

THE CONCEPT OF A BACKEND-FOR-FRONTEND



- [Microsoft.Extensions.AI libraries \(Preview\)](#) – Microsoft Learn
- [.NET Aspire Overview](#) – Microsoft Learn
- [The insecurity of Oauth 2.0 in frontends](#) – Phillipe de Ryck, NDC Security 2023
- [Securing SPAs and Blazor Applications using the BFF \(Backend for Frontend\) Pattern](#) – Dominick Baier, NDC Security 2023
- [Secure OIDC Client in Browser](#) – AXA France, GitHub Repository
- [Reading JSON and Binary Data from Multipart/Form data](#) – Andrew Lock's Blog

