avanade

Aplicações Modernas em MicroServiços com Ágil e DevOps

Andersson Pinheiro

@losttech

in/anderssonmelo

THE DEVELOPER'S CONFERENCE

Quem é a Avanade?

Consultoria **líder em soluções inovadoras de transformação digital**, modernização de TI e soluções de negócios.

Entregamos por meio do poder das nossas pessoas e do ecossistema Microsoft.

Digital Business and Modern IT

As defined by Gartner..

Digital Business Optimization

Digital Business Optimization Adds Significant Value Without Changing Business Models

Improve productivity and existing revenue	Better customer experience

Digital Business Transformation

Digital Business Transformation Leads to Net-New Revenue and Business Models

Net-new products and services	New business models

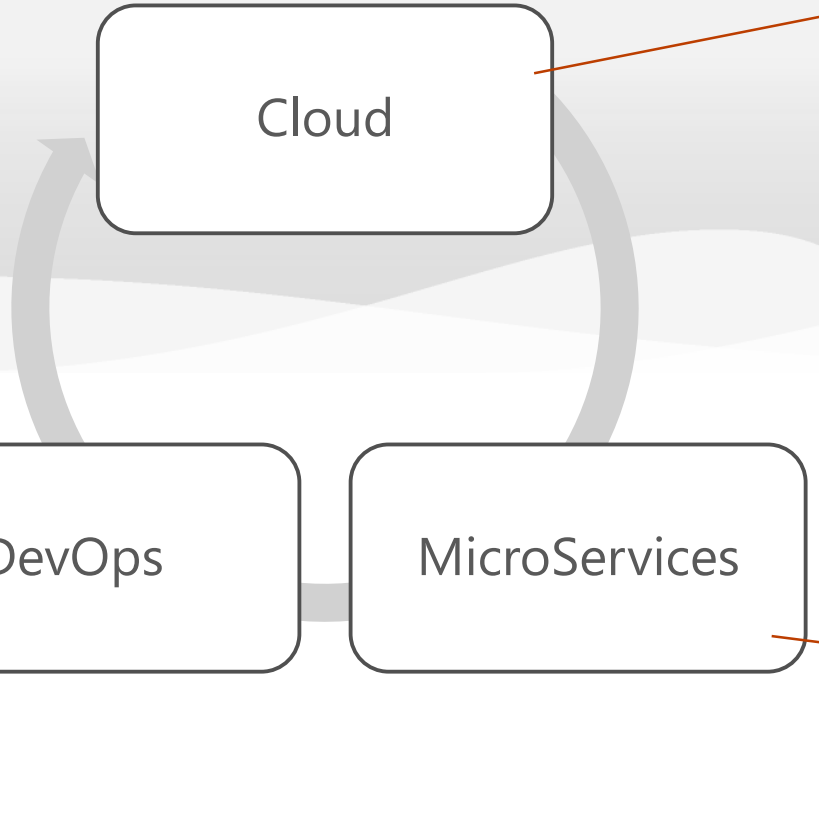
Modern IT

Be it optimization or transformation.
IT modernization is a prerequisite to any digital business initiatives.

avanade

"Digital Business Ambition: Transform or Optimize?" Gartner, June 2017

Desafios do Modern IT



Adotar *Cloud* em uma estratégia agnóstica, evitando o *vendor lock-in* através de **abstrações técnicas**

Aplicar DevOps tornando tudo isto possível através de CI,CD e CT – mas deve **suportar múltiplas clouds** e **lidar com múltiplos serviços**

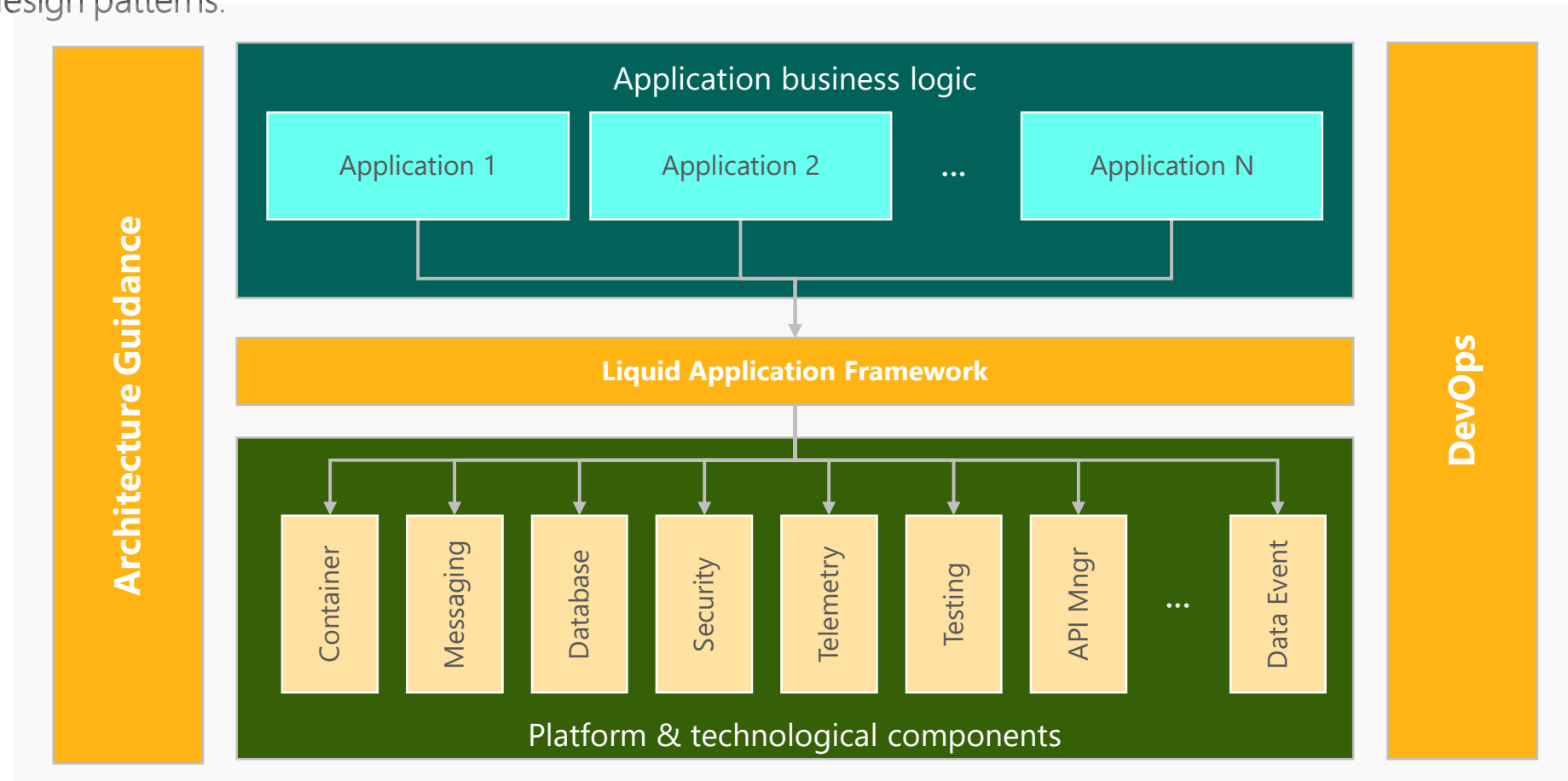
Divide suas aplicações por agregados de domínio, mas **aumenta a comunicação entre os pares**

Como lidamos com esta complexidade?

avanade

Enters *Liquid Application Framework*

Liquid compreende um **Orientação técnica de arquitetura**, um conjunto de components reutilizáveis de Código e uma esteira de DevOps pipeline **ready-to-use** para criar *Liquid Applications* que **desacoplam a lógica de negócios de uma aplicação** da camada de **componentes tecnológicos** ou de plataformas. É baseado na **lightweight architectures** e em **cloud design patterns**.

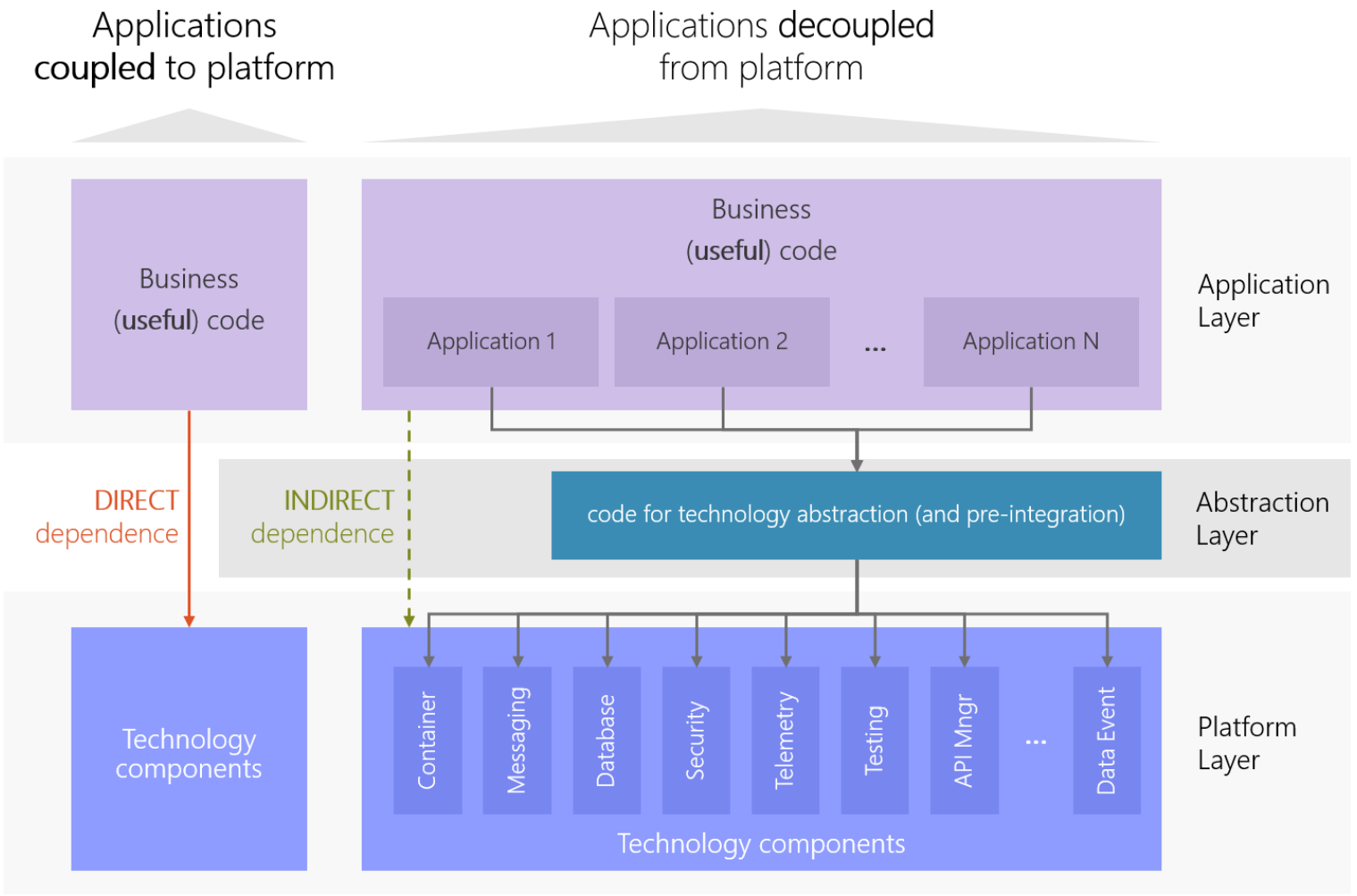


avanade

Camada Abstração Plataforma

A estratégia mais fundamental que orienta o design da *Liquid* é sua responsabilidade de **dissociar** o máximo possível os componentes do aplicativo dos componentes da **plataforma**.

O *Liquid* concentra a maioria das dependências específicas da plataforma (tecnologia) em um único ponto de código, liberando assim muitos componentes de aplicativos (comerciais) para acessar primitivas pré-integradas e de alta abstração para realizar seus trabalhos.

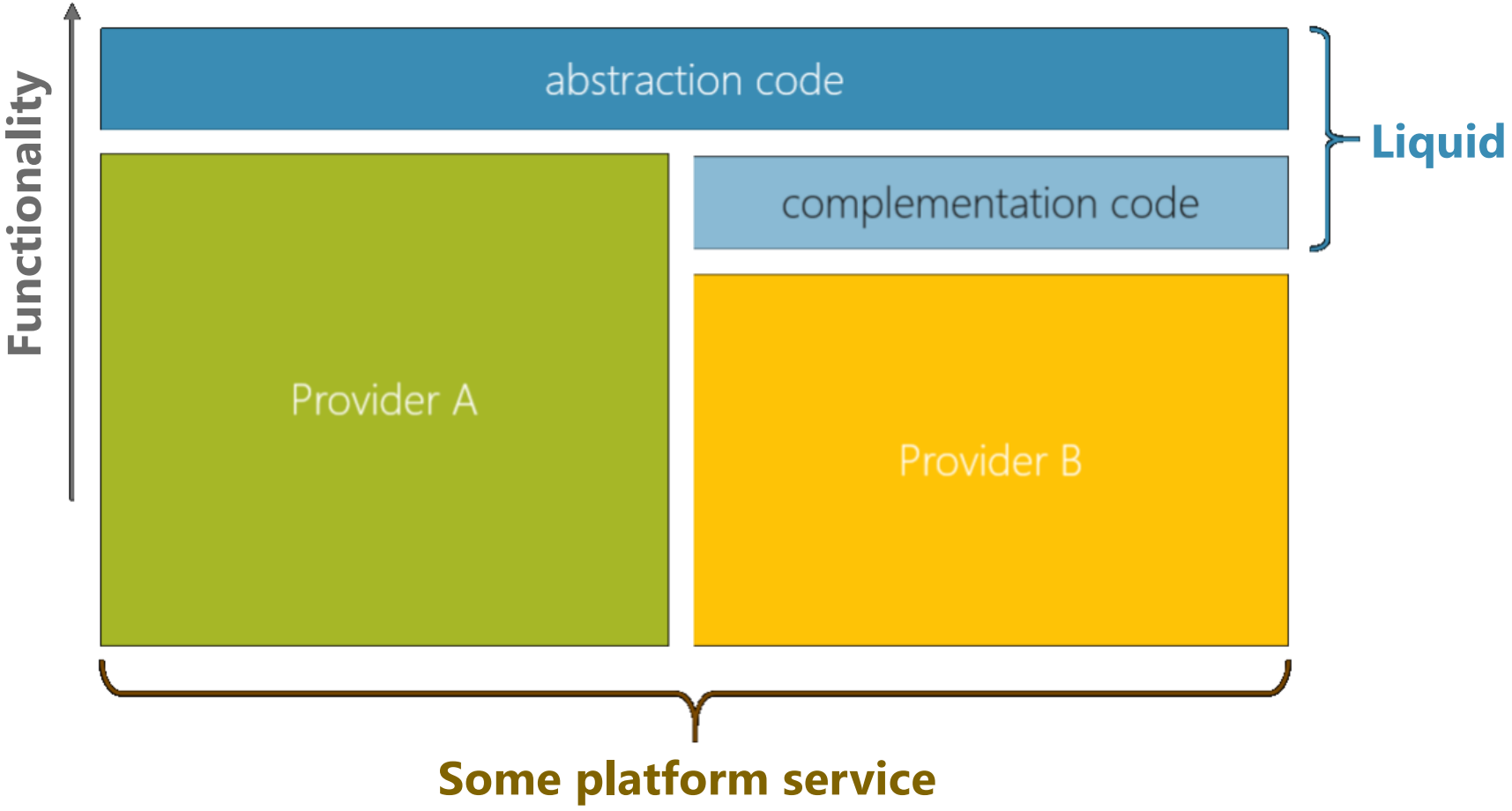


avanade

Nivelando fornecedores de plataforma

Além de abstrair os componentes de tecnologia, o *Liquid* também tenta **obter o melhor de todos os provedores**, configurando as primitivas de serviço (para componentes do aplicativo) no nível mais alto de suas funcionalidade.

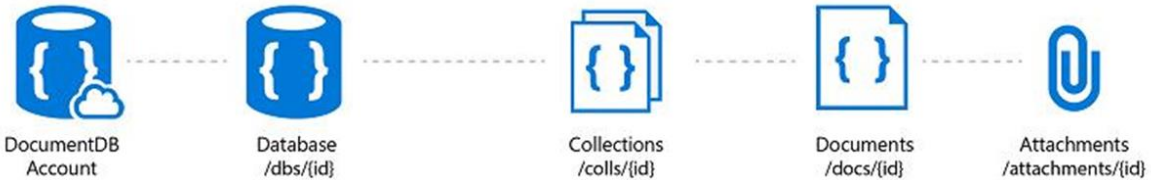
Para seguir essa estratégia e, ao mesmo tempo, poder **trocar componentes da plataforma**, o Liquid adiciona algum código complementar ao "**cartucho**" para um provedor com recursos mais pobres, de modo a elevá-lo ao alto nível geral de funcionalidade.



avanade

Nivelando fornecedores de plataforma

Exemplo:
Conceito do DB Attachments – substituindo **Azure CosmosDB** por **AWS DynamoDB** simplesmente alterando o **cartucho**




```
//Injects the Repository cartridge for Azure Cosmos DB
WorkBench.UseRepository<CosmosDB>();
...

//Repository has the concept of named binary files attached to an entity
var clientPicture = await Repository.GetAttachmentAsync(clientId, "profile.jpg");
```

```
//Injects the Repository cartridge for AWS Dynamo DB Repository
WorkBench.UseRepository<DynamoDB>();
...

//Repository STILL has the concept of named binary files attached to an entity
var clientPicture = await Repository.GetAttachmentAsync(clientId, "profile.jpg");
```

avanade

Separação da lógica de negócio (da lógica técnica)

Exemplo:
Recebendo uma mensagem de uma fila do Azure ServiceBus

With Liquid

```
...

//Injects the MessageBus cartridge for Azure Service Bus
WorkBench.UseMessageBus<ServiceBus>();

...

[MessageBus("ESHOP")]
    public class OrderingWorker : LightWorker
    {
        [Queue("order_to_ship")]
        public void ShipOrder(OrderToShipMessage message)
        {
            ValidateInput(message);

            //Calls domain (business) logic
            var response = Factory<OrderService>().Ship(message.MapTo<OrderToShipViewModel>());

            Terminate(response);
        }
```

Direct

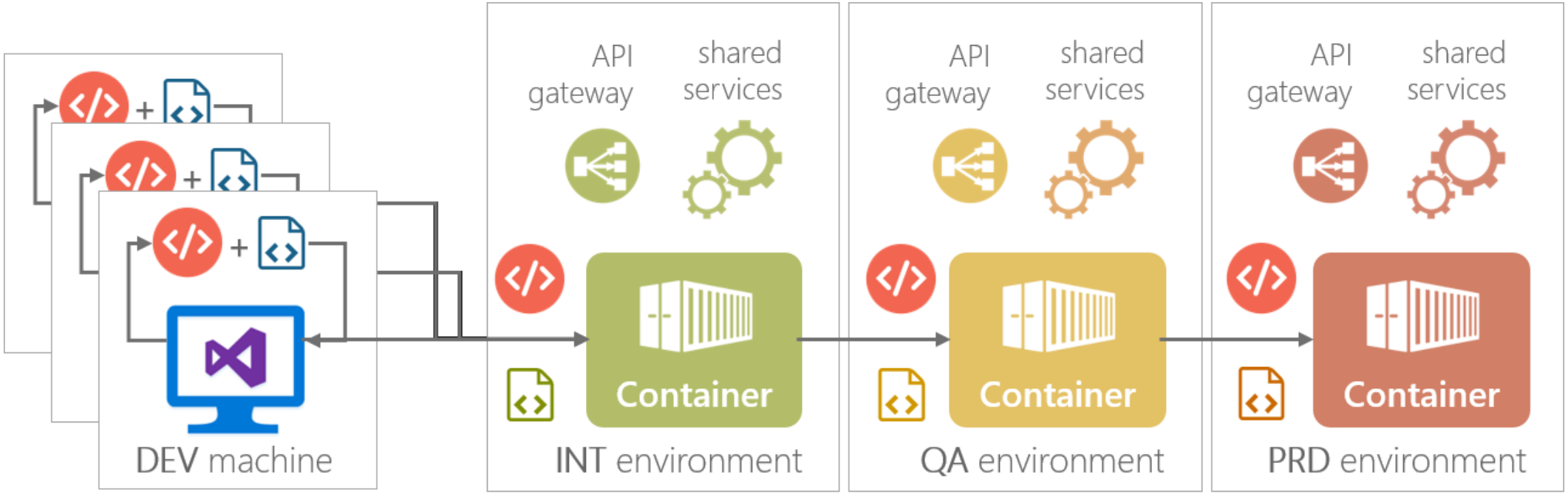
```
...

private void RegisterEventBus(IServiceCollection services)
        {
            var subscriptionClientName = Configuration["SubscriptionClientName"];

            if (Configuration.GetValue<bool>("AzureServiceBusEnabled"))
            {
                services.AddSingleton<IEventBus, EventBusServiceBus>(sp =>
                {
                    var serviceBusPersisterConnection = sp.GetRequiredService<IServiceBusPersisterConnection>();
                    var iLifetimeScope = sp.GetRequiredService<ILifetimeScope>();
                    var logger = sp.GetRequiredService<ILogger<EventBusServiceBus>>();
                    var eventBusSubcriptionsManager = sp.GetRequiredService<IEventBusSubscriptionsManager>();

                    return new EventBusServiceBus(serviceBusPersisterConnection, logger,
                        eventBusSubcriptionsManager, subscriptionClientName, iLifetimeScope);
                });
            }

            ...

            RegisterEventBus(services);

            //create autofac based service provider
            var container = new ContainerBuilder();
            container.Populate(services);

            return new AutofacServiceProvider(container.Build());
}

...

 private void ConfigureEventBus(IApplicationBuilder app)

        {
            var eventBus = app.ApplicationServices.GetRequiredService<IEventBus>();

            ...

            eventBus.Subscribe<OrderStatusChangedToShippedIntegrationEvent, OrderStatusChangedToShippedIntegrationEventHandler>();
        }

...

    public class OrderStatusChangedToShippedIntegrationEventHandler : IIntegrationEventHandler<OrderStatusChangedToShippedIntegrationEvent>
    {
        private readonly IHubContext<NotificationsHub> _hubContext;

        public OrderStatusChangedToShippedIntegrationEventHandler(IHubContext<NotificationsHub> hubContext)
        {
            _hubContext = hubContext ?? throw new ArgumentNullException(nameof(hubContext));
        }

        public async Task Handle(OrderStatusChangedToShippedIntegrationEvent @event)
        {
        ...
        }
```

avanade

DevOps for Microservices

O mesmo código deve se comportar da mesma maneira no ambiente de desenvolvimento independente e nos compartilhados remotos, com apenas alterações em seus arquivos de configuração.

O Liquid facilita a tarefa, lidando com essas alterações de configuração, controlando a maneira como um microsserviço se comunica externamente, consultando ou sendo consultado.



avanade

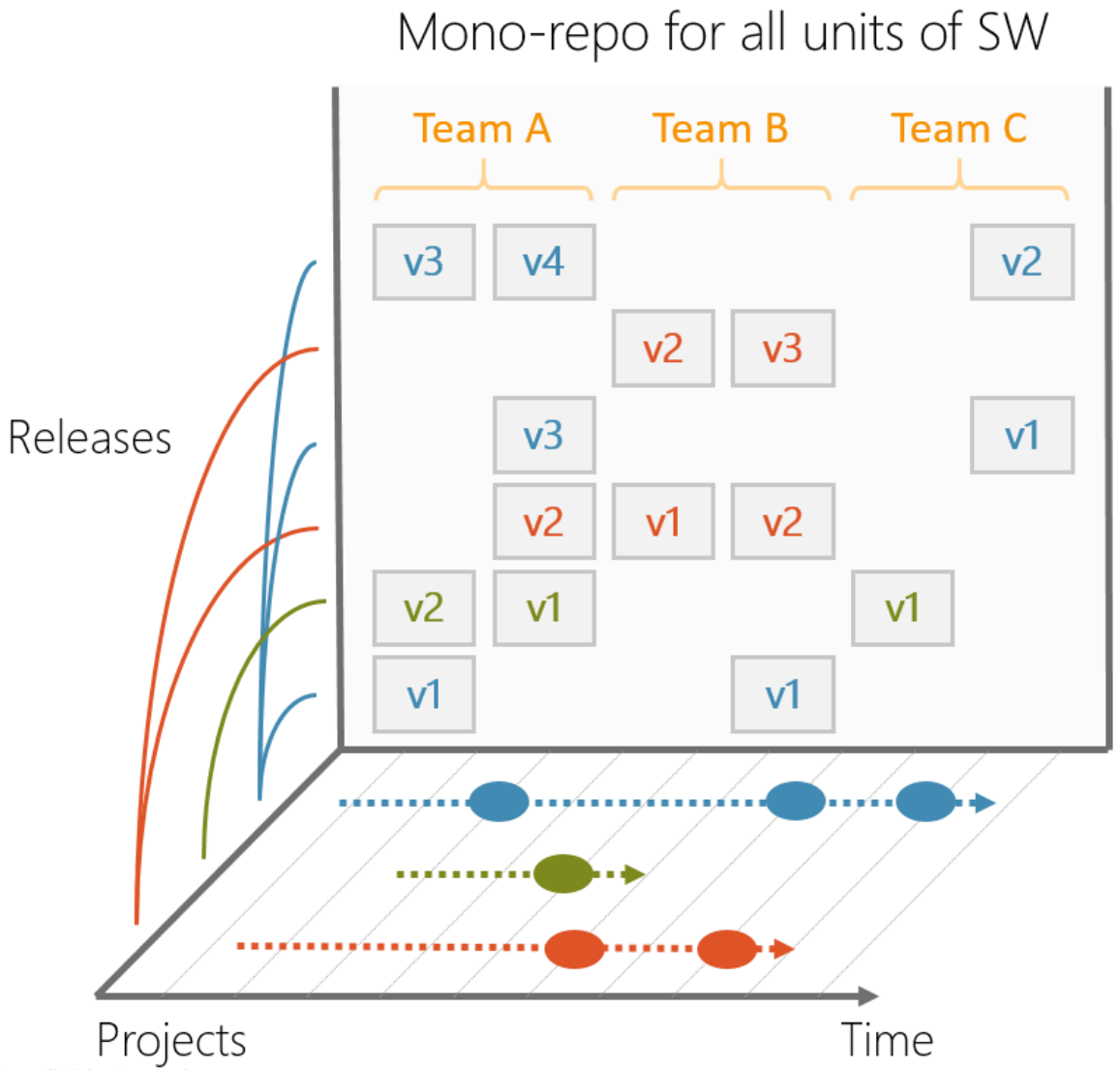
DevOps for Microservices

Como tratar de centenas de unidades de software para serem desenvolvidas, implantadas e executadas individualmente?

Portanto, os microsserviços devem ter CI, CD e CT independentemente, iniciando com um delta de alteração de código que foi feita como uma *release* de um dos projetos ativos - todos compartilhando o mesmo repositório de código em todos os sentidos.

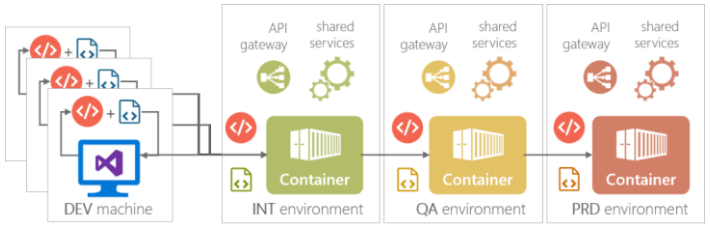
Liquid fornece templates de CI, CD e CT para trabalhar essa estratégia, começando pequeno (poucos microsserviços) mas tornando a adição de novos uma questão de simplesmente adicionar novo código. E isso pode aumentar para centenas ou milhares.

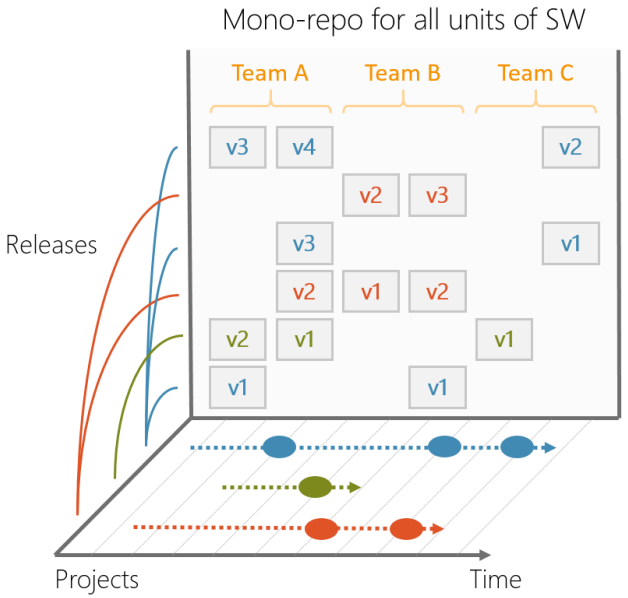
O novo código é implantado automaticamente pelo mesmo pipeline de DevOps. Não há necessidade de novas definições de build para cada novo microsserviço.

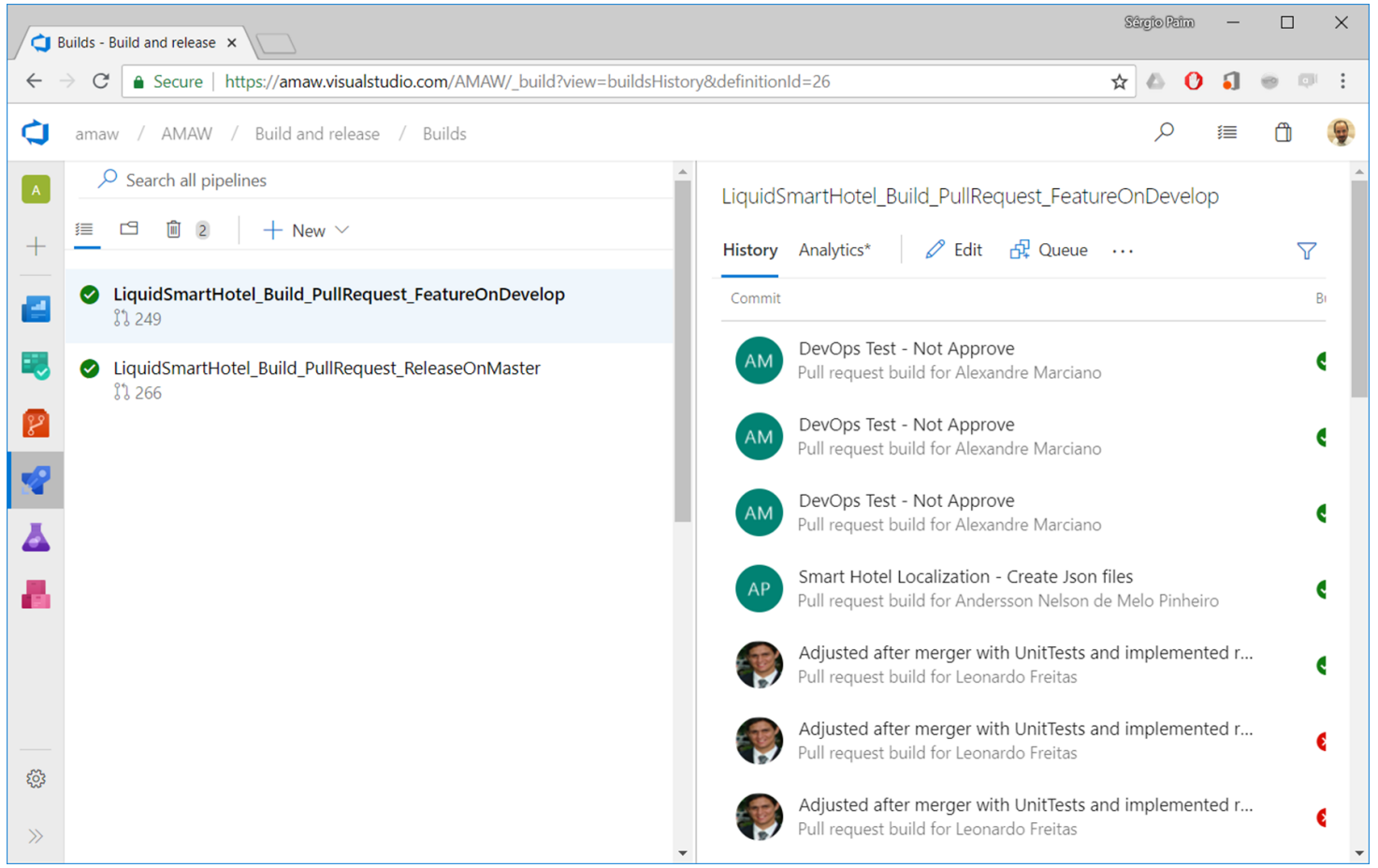


avanade

DevOps for Microservices





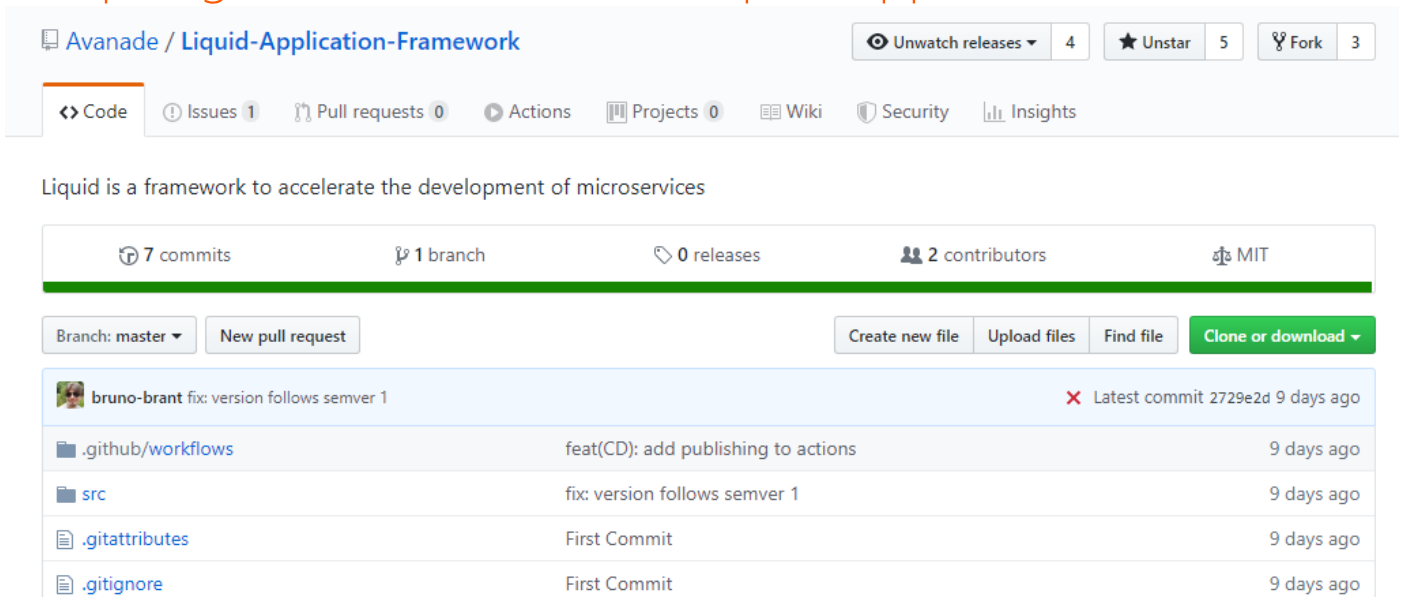


avanade

Open Source Strategy

O Liquid foi desenvolvido com a intenção de ser disponibilizado para a comunidade de forma opensource e uma versão prerelease já está disponível na conta github da Avanade.

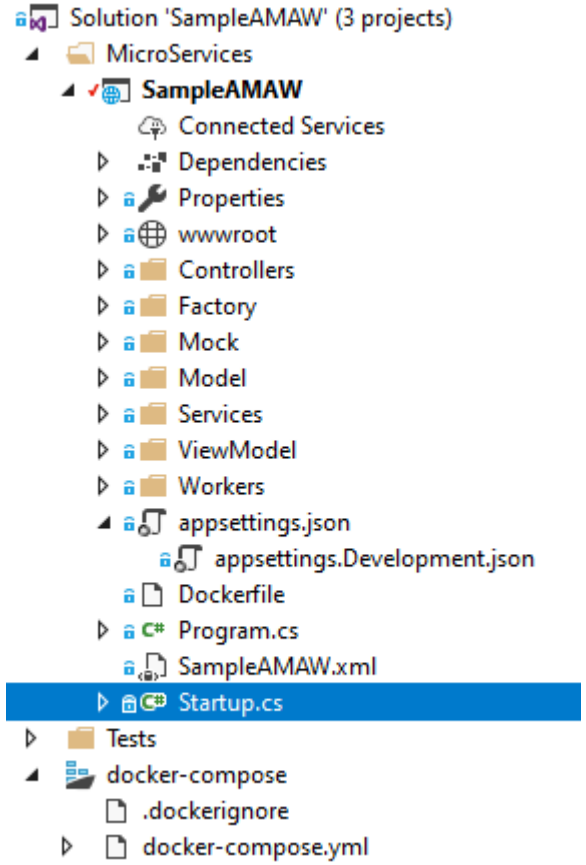
https://github.com/Avanade/Liquid-Application-Framework



avanade

Liquid Application Framework

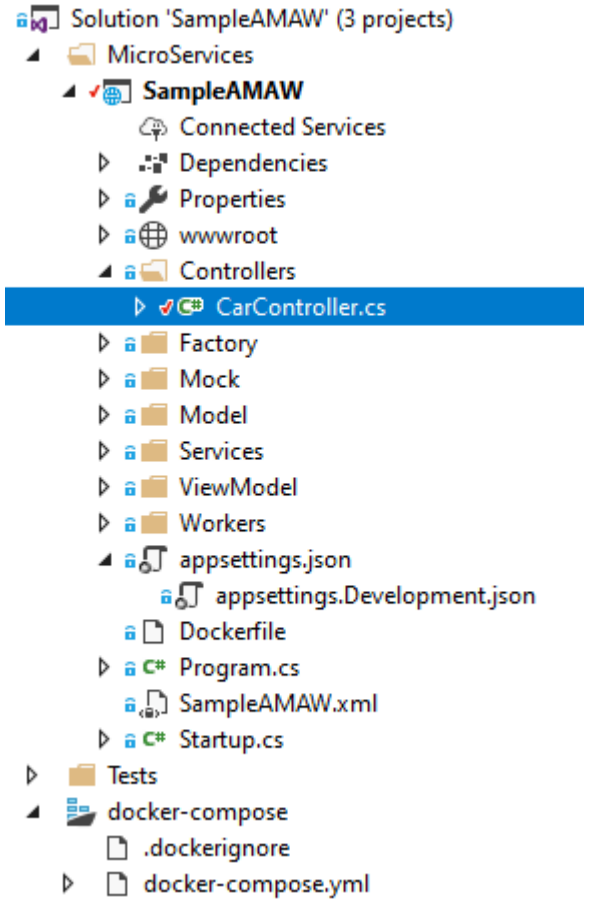
```
/// <summary>
/// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
/// </summary>
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    //Start the configuration on WorkBench
    WorkBench.UseTelemetry<AppInsights>();
    WorkBench.UseRepository<CosmosDB>();
    WorkBench.UseMessageBus<ServiceBus>();

    app.UseWorkBench();

    app.UseMvc();
}
```



avanade

Liquid Application Framework

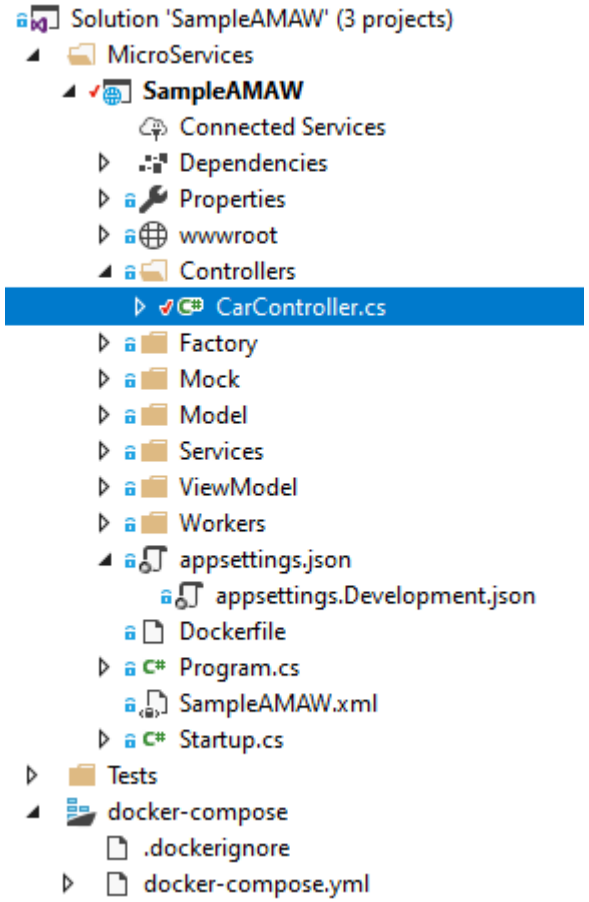
```
/// <summary>
/// Class responsible for exposure of Car records
/// </summary>
[ApiVersion("1")]
[Route("api/v{version:apiVersion}/[controller]")]
public class CarController : LightController
{
    /// <summary>
    /// Get All Records
    /// </summary>
    /// <remarks>
    /// GET api/v1.0/Car
    /// </remarks>
    [HttpGet]
    [ApiExplorerSettings(GroupName = "v1")]
    public async Task<IActionResult> GetAll()
    {
        var data = await Factory<CarService>().GetAllAsync();
        data.ViewModelData = data.ModelData;
        return Result(data);
    }
```



avanade

Liquid Application Framework

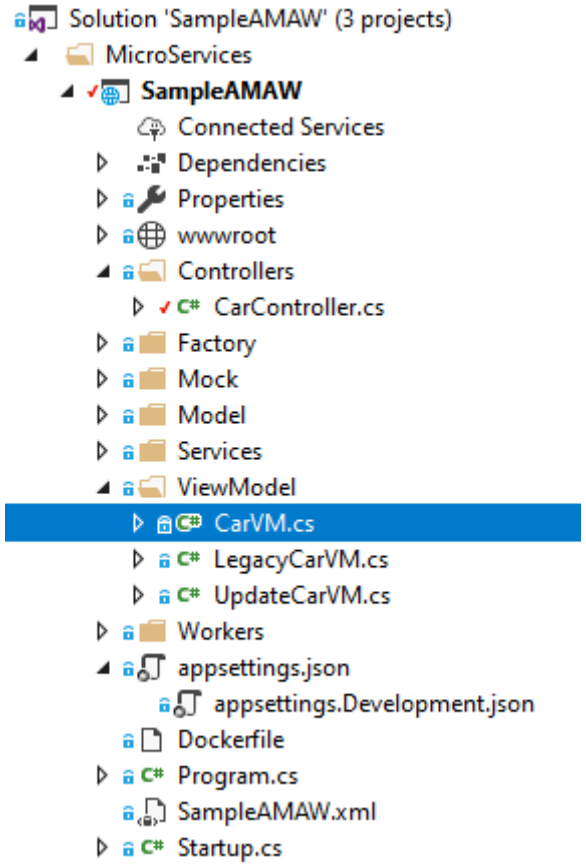
```
/// <summary>
/// Post Record
/// </summary>
/// <remarks>
/// Post api/v1.0/Car
/// </remarks>
[HttpPost]
[ApiExplorerSettings(GroupName = "v1")]
public async Task<IActionResult> PostAsync([FromBody] CarVM viewModel)
{
    ValidateInput(viewModel);

    var data = await Factory<CarService>().SaveAsync(viewModel);
    data.ViewModelData = data.ModelData;
    return Result(data);
}
```



avanade

Liquid Application Framework

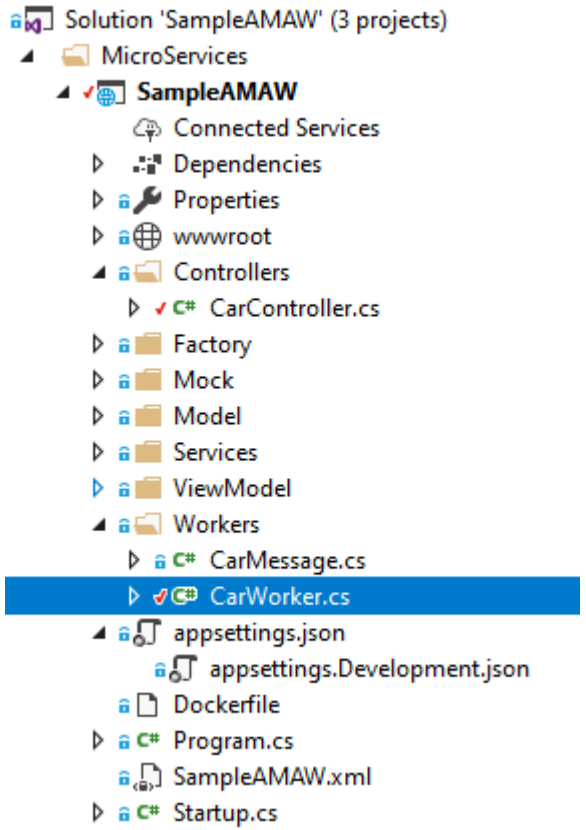
```
/// <summary>
/// Class CarVM
/// </summary>
public class CarVM : LightViewModel<CarVM>
{
    /// <summary>
    /// Car VM Description
    /// </summary>
    public string Description { get; set; }

    /// <summary>
    /// Car VM Description
    /// </summary>
    public string ModelYear { get; set; }

    /// <summary>
    /// Method to validate a Car VM Id
    /// </summary>
    public override void Validate()
    {
        RuleFor(x => x.Description).NotEmpty().WithErrorCode("DESCRIPTION_MUSTNOT_BE_EMPTY");
        RuleFor(x => x.ModelYear).NotEmpty().WithErrorCode("MODELYEAR_MUSTNOT_BE_EMPTY");
    }
}
```



avanade

Liquid Application Framework

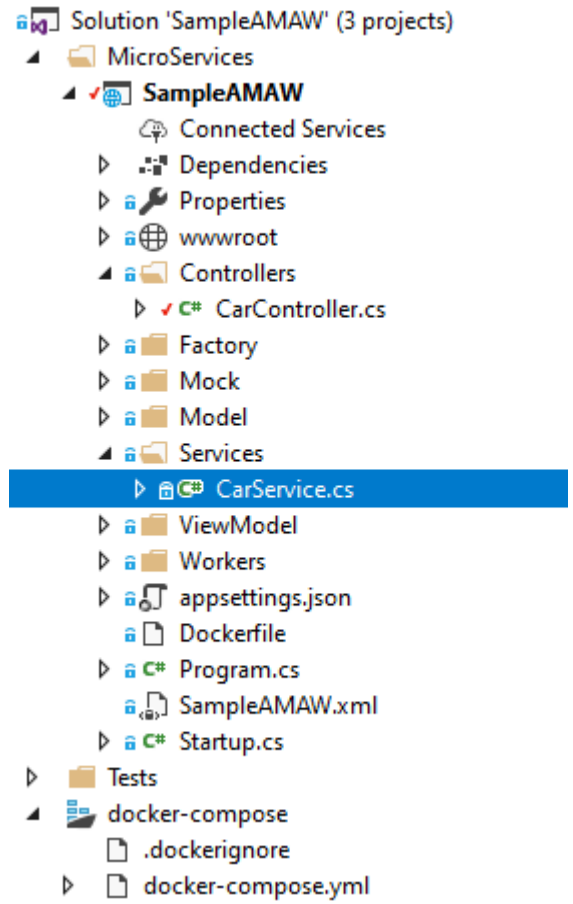
```
/// <summary>
/// Worker of Car
/// </summary>
[MessageBus("BUS")]
public class CarWorker : LightWorker
{
    /// <summary>
    /// Method Get Message
    /// </summary>
    /// <param name="message">CarMessage Object</param>
    [Topic("car", "LegacyCarSubs", 10, false)]
    public void GetMessage(CarMessage message)
    {
        ValidateInput(message);

        //Calls domain (business) logic
        var response = Factory<CarService>().SaveWorker(message.LegacyCarVM);

        //Terminates the message according to domain response
        Terminate(response.Result);
    }
}
```



avanade

Liquid Application Framework

```
/// <summary>
/// Services of Car
/// </summary>
public class CarService : LightService
{
    /// <summary>
    /// Get a record of Car
    /// </summary>
    /// <param name="id">Car Id</param>
    /// <returns></returns>
    public async Task<DomainResponse> GetAsync(string id)
    {
        Telemetry.TrackEvent("Get Record");
        Car records = await Repository.GetByIdAsync<Car>(id);
        return Response(records);
    }

    /// <summary>
    /// Save a record of Car
    /// </summary>
    /// <param name="viewModel"></param>
    /// <returns></returns>
    public async Task<DomainResponse> SaveAsync(CarVM viewModel)
    {
        Car model = new Car();
        model.MapFrom(viewModel);
        Telemetry.TrackEvent("Save Record");
        var records = await Repository.AddOrUpdateAsync(model);
        return Response(records);
    }
```

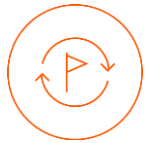


avanade

Wrap up

Benefícios

O *workbench* aumenta a produtividade e qualidade do desenvolvimento e operação de aplicações corporativas por reduzir sua complexidade através do uso de components técnicos pré-construídos e pré-integrados.

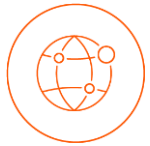


Focus on your business

Liquid abstracts technical complexity from business software developers, enhancing their productivity, quality and helping them to focus on useful code.

Level up platform providers

Ensuring top functionality in the market even if a particular vendor has not yet reached the level of the others.



Catalyze the use of delivery centers

By getting most of the technical design out of the box, remote teams can focus communication on the requirements (backlog) and on the design of the business solution (application layer).

Avoid vendor lock-in without losing functionality

Liquid puts itself between application (business) components and underlying platform (technical) components to avoid vendor and technology lock-in. Hence there is a way of simply replacing *Liquid* "cartridges" of each type from one specific technology to another.

Faster project setup

Use pre-built *Liquid* templates to start building your business application.

Scale

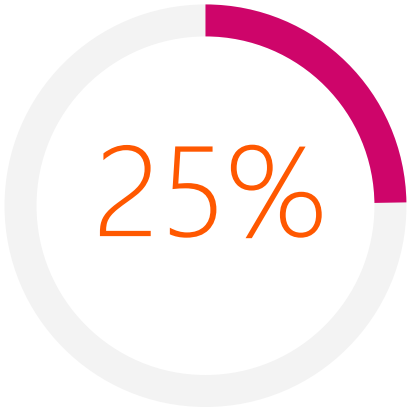
Liquid is built on top of container technologies, enabling your business to scale as needed.

avanade

Benefits

Microsserviços menores

Um microsserviço tradicional poderia ser feito com quase 80% menos código. Todo o código usado para controlar, integração, tratamento de erros, conexões e autenticação é encapsulado e com garantia de qualidade, permitindo que você se concentre nos seus negócios.

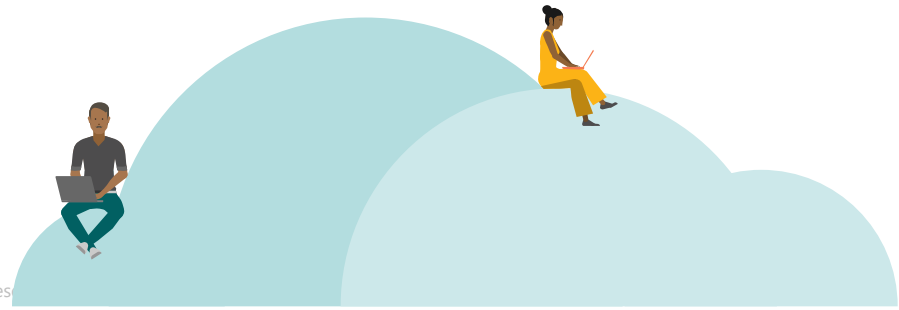


Reduz custo

Usando o Liquid o custo do seu projeto pode ser reduzido em 25%.

DevOps pronto

Preparado para CD, CI and CT garantir entregas mais rápidas e confiáveis.



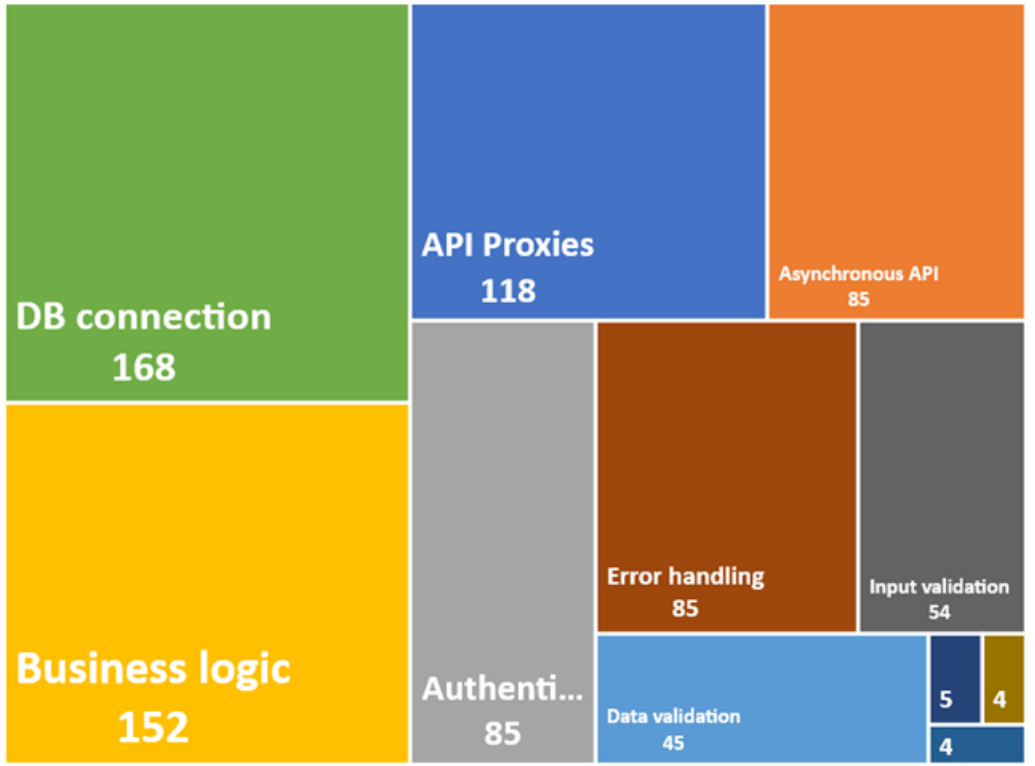
avanade

Impacto na productividade (e qualidade)

With AMAW (139 lines)

- API Proxies
- Asynchronous API
- Authentication
- Business logic
- Data validation
- DB connection
- Dependency Injection
- Error handling
- Input validation
- Synchronous API
- Telemetry

Direct (805 lines)



avanade

Liquid Application Framework aborda os desafios da Transformação Digital

- Conquiste *time-to-value* com velocidade
- Colete métricas e *insights* de clientes
- Aprimore a confiança em *outsource*
- Seja mais produtivo
- Evite o aprisionamento do fornecedor sem perder funcionalidades
- Adote uma estratégia *multi-cloud*

avanade