



# Nome: Leonardo Costa de Carvalho

Neste documento, aproveite para registrar as respostas das atividades formativas que serão apresentadas durante as semanas.

## **Atividade formativa: Hora de exercitar!**

### **Respostas da semana 3:**

Essa proposta de uma arquitetura em computação em nuvem precisa equilibrar escalabilidade, disponibilidade, desempenho por se tratar de um fluxo macro do sistema de matrícula online, que envolve um grande volume de inscrições em períodos sazonais críticos.

No processamento escalável, o ideal é usar o AWS EC2 com Auto Scaling, Google Compute Engine, ou Azure Virtual Machine Scale Sets. Outra alternativa serverless poderia ser o AWS Lambda ou Azure Functions para partes do sistema como validações, envio de notificações, etc.

Para um banco de dados gerenciado seria recomendado o Amazon RDS (PostgreSQL ou MySQL), Cloud SQL (GCP), ou Azure Database. Na parte de documentos, o Amazon S3 / Azure Blob Storage para uploads (PDFs, RG, etc.).

Caso seja dados semi-estruturados, pensamos no MongoDB Atlas (NoSQL). Para distribuir tráfego entre servidores durante picos de acesso, podemos pensar no AWS Elastic Load Balancer, GCP Load Balancer. Para usar o CDN (Content Delivery Network), que irá acelerar o carregamento do sistema web, especialmente documentos e imagens como o CloudFront (AWS), Cloudflare, Azure CDN.

No caso de fila de mensagens e orquestração, temos como exemplo o Amazon SQS, Google Pub/Sub, Azure Service Bus — para tratar eventos assíncronos como validação de documentos.

No monitoramento e logging, existe o CloudWatch, Azure Monitor ou GCP Operations para monitoramento de desempenho e alertas de falhas.

Já para o armazenamento de arquivos, iremos fazer o upload de documentos dos estudantes em buckets com controle de acesso (ex: Amazon S3 com políticas IAM).

Será necessário a autenticação e autorização de serviços gerenciados como Firebase Auth, AWS Cognito ou Azure AD B2C.

Para o deploy contínuo (CI/CD), o mais comum são os pipelines com GitHub Actions, GitLab CI, ou ferramentas como AWS CodePipeline, Azure DevOps.

Como dimensionar a arquitetura de nuvem? Com modelo de microserviços, ao separar o sistema por funcionalidades (ex: módulo de matrícula, módulo de validação, módulo de contrato), ou facilidade de escalar somente partes críticas, como validação e pagamento. No Auto Scaling horizontal, teremos que configurar regras para adicionar instâncias durante picos de acesso e removê-las após. No caso de armazenamento e banco com escalabilidade vertical e horizontal, o read replicas em banco de dados para aliviar leitura e particionamento (sharding) se necessário no futuro.



Alta disponibilidade será com instâncias distribuídas em múltiplas zonas (multi-AZ), backups automáticos e failover. Usando também o cache com redis/Memcached para evitar consultas repetidas no banco de dados (ex: verificação de status da matrícula).

A Justificativa se dá por ser:

Escalável: Atende picos de 20.000 inscrições, escalando sob demanda.

Alta disponibilidade: Imprescindível para não interromper o processo de matrícula em períodos críticos.

Flexível: Permite iniciar com menor estrutura e crescer conforme uso.

Custo otimizado: Recursos como serverless, bancos gerenciados e CDN reduzem custos em períodos de baixa demanda.

Performance: Balanceamento de carga, cache e filas de processamento garantem fluidez mesmo com alta carga.

Infraestrutura robusta desde o início ou escalável por demanda, será recomendado começar com infraestrutura escalável e modular. Módulos essenciais com capacidade mínima garantida (ex: banco, autenticação, upload), auto Scaling habilitado para picos e monitoramento ativo para adaptar conforme uso.

Vou pegar como referência os sistemas de matrícula do ENEM ou SISU (INEP), que utilizam arquitetura elástica com平衡adores e escalabilidade automática, além das matrículas do Coursera / edX que também segue o padrão de microserviços com armazenamento em nuvem e autenticação gerenciada.

#### **Respostas da semana 4:**

Nossa camada de apresentação (Frontend) utilizaria neste caso o Azure Blob Static Web + Azure CDN, para hospedagem de frontend web (React/Vue/Angular), onde o estudante faz a matrícula.

Na camada de API (Backend), serviço de API Gateway para receber as requisições REST. Lambda Functions (Serverless), para endpoints na criação de matrícula, validação de pagamento, validação de documentos, geração de contrato, notificações ao estudante, pois, o serverless reduz custo em períodos de baixa demanda.

Na camada de processamento assíncrono, o ideal é usar o AWS Step Functions para orquestrar o fluxo entre as funções Lambda (opcional). Como estamos utilizando apenas o Azure, então fica como uma alternativa.

Na camada de banco de dados, utilizamos o Azure CLI, que realiza a conexão diretamente pelo navegador de internet, dentro do portal do Azure, sem a necessidade de usar a chave privada. Essa estrutura permite acompanhar o fluxo completo desde a solicitação até o pagamento. A infraestrutura com VMs no Azure

- VM 1: Banco de Dados – com SQL Server ou PostgreSQL.

- VM 2: API de Backend – serviços em Node.js, .NET ou outra stack para processar regras de crédito.



- VM 3: Aplicação Web/Administração – gerencia usuários, dashboards, aprovações.

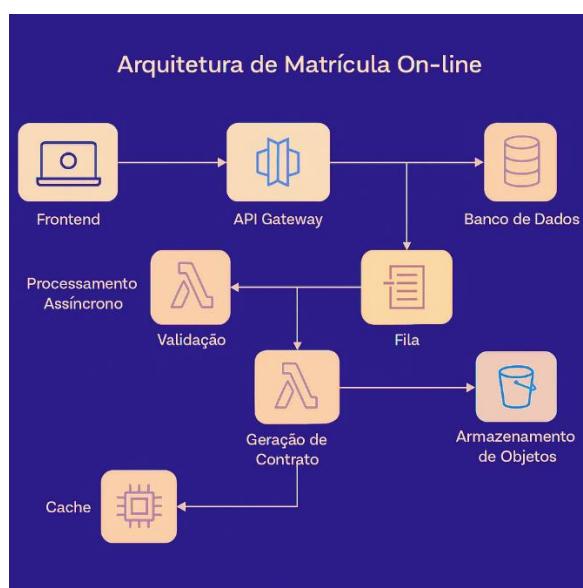
Você usaria:

- Azure Virtual Network (VNet) para isolar as VMs.
- Discos gerenciados com criptografia para proteger os dados.
- Azure Backup e Recovery Vault para segurança e resiliência.

Como alternativa gerenciada no Azure se quiser simplificar:

- Azure SQL Database (em vez da VM para o banco)
- Azure App Service (para backend e front)
- Azure Logic Apps ou Functions (para processar eventos automáticos, como gerar parcelas ou cobranças)
- Azure Key Vault para armazenar segredos, chaves e conexões com segurança.

Os cuidados para não estourar o orçamento (evitar queimar os US\$ 100 rapidamente) seria o uso de Lambda (Serverless) só paga por execução, usando instâncias pequenas ou Free Tier ou S3 com baixo custo por armazenamento.



### Respostas da semana 5:

Os alertas que serão configurados:

- Alertas críticos com ação em até 5 minutos usarão o recurso, a condição e a ação. Com o App Service / Function App com tempo de resposta médio maior que 2 segundos por 5 min, na escalada horizontalmente haveria um aumento na memória. Com o SQL Database, DTU (ou vCore utilization) maior que 80% por 5 minutos, numa escalada tier do banco. Teria um queue Storage com número de mensagens na fila maior



que 10.000 (sinal de backlog). Criaria novas instâncias de consumidores (Functions) Storage Account, falha de upload ou timeout frequente para checar IOPS e throughput.

- Alertas de Capacidade de ação em até 1 hora, com o Blob Storage e uso de armazenamento maior que 80% da cota, começa a avaliar aumento de quota ou Limpeza de arquivos antigos. No caso de SQL Database, o espaço usado maior que 85% começa a provisionar mais espaço ou realizar manutenção.
- Alertas de custo de ação em até 12 horas, o orçamento mensal Azure com consumo maior que 70%, 90% e 100% começa a revisar a escalabilidade e ajustar o Auto Scaling para negociar orçamento.
- Alertas de Segurança usa WAF ou Firewall, em caso de tentativas de ataque (excesso de 4xx/5xx) ele ativa regras de bloqueio e revisa IPs suspeitos.
- Plano de ação por criticidade em nível alto vai até 5 minutos, após isso começa a usar DevOps + Operações em até 1 hora com ação corretiva (ex: escalar recursos) e revisão pela equipe de arquitetura. Em nível médio em até 12 horas, começa os ajustes preventivos, planejamento de melhorias na próxima janela de manutenção. Em caso de nível baixo em até 24 horas, haverá um monitoramento, replanejamento threshold de alerta se necessário.
- Próximas ações práticas no portal Azure é a configuração do Azure Monitor + Action Groups, para criar alertas para CPU, memória, filas e banco, criar Dashboard com application insights, ativar diagnostics logs para App Services e SQL e ativar Azure Cost Management para controlar gastos.





## Respostas da semana 6:

- Segurança de rede (Network Security) usando App Services / Function Apps, habilitação e restrição por IP e VNet Integration e impedir acessos de redes externas não autorizadas. Além do SQL Database para ativar Firewall de SQL, liberar apenas IPs de App Service e DevOps e evitar acessos externos. Uso do Blob Storage para restringir acesso com Private Endpoints + SAS Token para uploads e prevenir acesso público indevido aos documentos dos alunos. Uso também do API Gateway / Frontdoor, configurar WAF (Web Application Firewall) com regras de OWASP para bloquear ataques como SQL Injection, XSS, DDoS.
- Segurança de identidade e acesso (IAM e RBAC), acesso ao Azure Portal para implementar Azure AD MFA (Multi-Factor Authentication) e garantir que somente usuários verificados acessem o DevOps e CI/CD com permissões de acesso baseadas em papéis (RBAC), com principle of least privilege. Só vai permitir que cada equipe veja/edita o necessário, além do Storage Access ao configurar Managed Identity para que as funções Lambda/Functions acessem o Blob Storage sem chaves explícitas e evitar senhas em código. Uso do API Calls com proteção do Azure API Management + OAuth 2.0 ou Azure AD B2C autenticando todas as chamadas de API.
- Proteção de dados (Data Protection) com SQL Database para ativar Transparent Data Encryption (TDE) e garantir criptografia em repouso. Uso do Blob Storage para habilitar Encryption at Rest (Azure-managed keys) e proteger documentos enviados, além de tráfego API para forçar HTTPS/TLS 1.2 ou superior à proteger dados em trânsito.
- Auditoria e logs de segurança usando Azure Monitor e Log Analytics, e ativar o Diagnostic Logs em todos os serviços conseguindo assim auditar tentativas de acesso não autorizado. Também o Azure Security Center (Defender for Cloud) ao habilitar com nível Standard para monitoramento contínuo de ameaças.
- Estratégia de backup e recuperação com SQL Database e backups automáticos (7 a 30 dias de retenção) com restauração rápida em caso de falha ou ataque junto ao Blob Storage e Soft Delete ativados para recuperar documentos deletados por engano.

Deve ser feito uma lógica da abordagem de segurança, estratégia desenhada com foco em:

- Defesa em Camadas (Defense in Depth)
- Menor Privilégio (Least Privilege)
- Monitoramento Contínuo
- Proteção de Dados sensíveis (Privacy by Design)
- Escalável, mas seguro



## Respostas da semana 7:

Começamos neste tópico o plano de redundância com a ampliação do Multi-Region / Multi-AZ. Com o uso do App Service e Azure SQL Database em região primária com failover para outra zona de disponibilidade (AZ). E mais o Storage e Queue com redundância geográfica (GRS - Geo-Redundant Storage).

Depois da ampliação, o uso do Multi-Region Active-Passive (Disaster Recovery), uma réplica do ambiente será mantida em outra região Azure. Replicação de dados quase em tempo real (com Azure Geo-Replication nos bancos). Plano de failover manual ou automático usando Azure Traffic Manager. O plano aqui é minimizar downtime em caso de falha regional ou de data center.

Sobre a escalabilidade, seria necessário um modelo de escalonamento, com escalabilidade horizontal, Auto Scaling de App Service Plan e Functions com base em CPU, memória e filas, para suportar picos repentinos (ex: abertura de matrículas). Escalabilidade Vertical (quando necessário) com upgrade de tier de banco e storage durante picos para garantir performance durante períodos críticos.

Banco de dados e Read Replicas ou Azure Hyperscale para aliviar carga de leitura. Adotar Auto-Scaling baseado em métrica (CPU, throughput de fila, número de requisições) para manter o desempenho sem desperdiçar orçamento.

No caso do Backup, é importante o uso do SQL Database, backups automáticos do Azure, backup full diário + logs transacionais de 7 à 35 dias. Blob Storage (documentos dos alunos), Soft Delete + versionamento + snapshots em até 30 dias, com configurações de infraestrutura e exportação de templates ARM e pipelines de forma semanal dentro do prazo de 30 dias.

Verificação de integridade pode ser feito com o restore trimestrais em ambiente de homologação, sem deixar o Disaster Recovery (plano simplificado), RPO (Recovery Point Objective) com no máximo 15 minutos de perda de dados (banco e documentos) e RTO (Recovery Time Objective) com recuperação completa em até 1 hora. Outra estratégia é a replicação entre regiões (Geo-Redundant), failover planejado, scripts automatizados para restaurar infraestrutura via Azure Resource Manager (ARM Templates) e testes de DR com simulações a cada 6 meses.

No caso das mudanças para atender a nova demanda (10x mais usuários), migração para Azure SQL Hyperscale ou Azure Cosmos DB (se decidir por NoSQL para partes do sistema), migração de Azure Queue para Azure Service Bus Premium e funções / backend para reforçar o uso de Azure Kubernetes Service (AKS) ou App Service Premium para mais controle sobre auto-escalonamento.

Agora o sistema:

- É resiliente
- Suporta escalabilidade horizontal e vertical
- Tem um plano claro de Disaster Recovery
- Tem redundância geográfica
- Está pronto para a nova demanda (10x mais público)
- Mantém custo controlado fora de pico, com crescimento elástico conforme uso.