Utilizando um microsserviço para comunicar com recursos externos, a anatomia de um gateway node.

Cassiano Raimar Silva

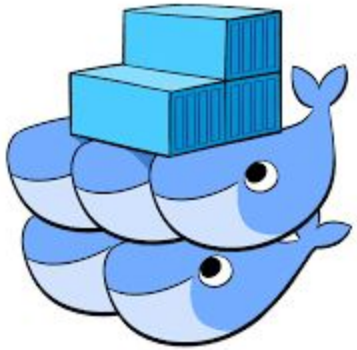
Contextualizando

Contextualizando

- minutrade
  - Engajar e reconhecer as pessoas com a melhor recompensa, promovendo inclusão digital e social.
- Integração com alianças

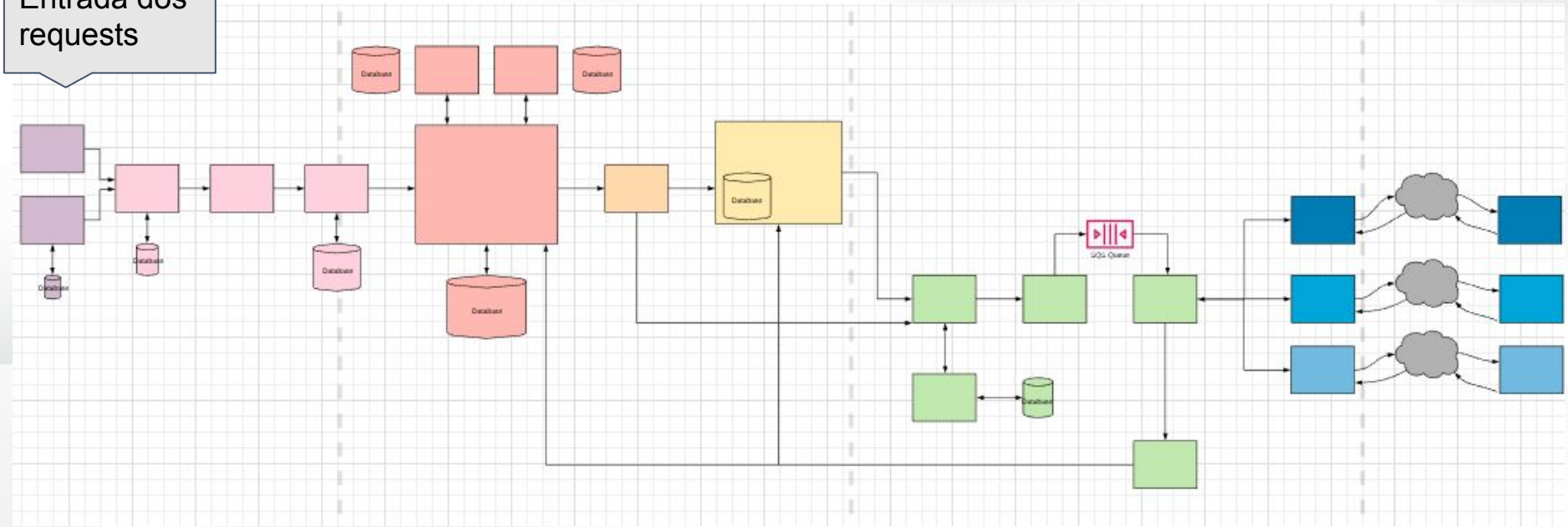
Contextualizando

- O que utilizamos

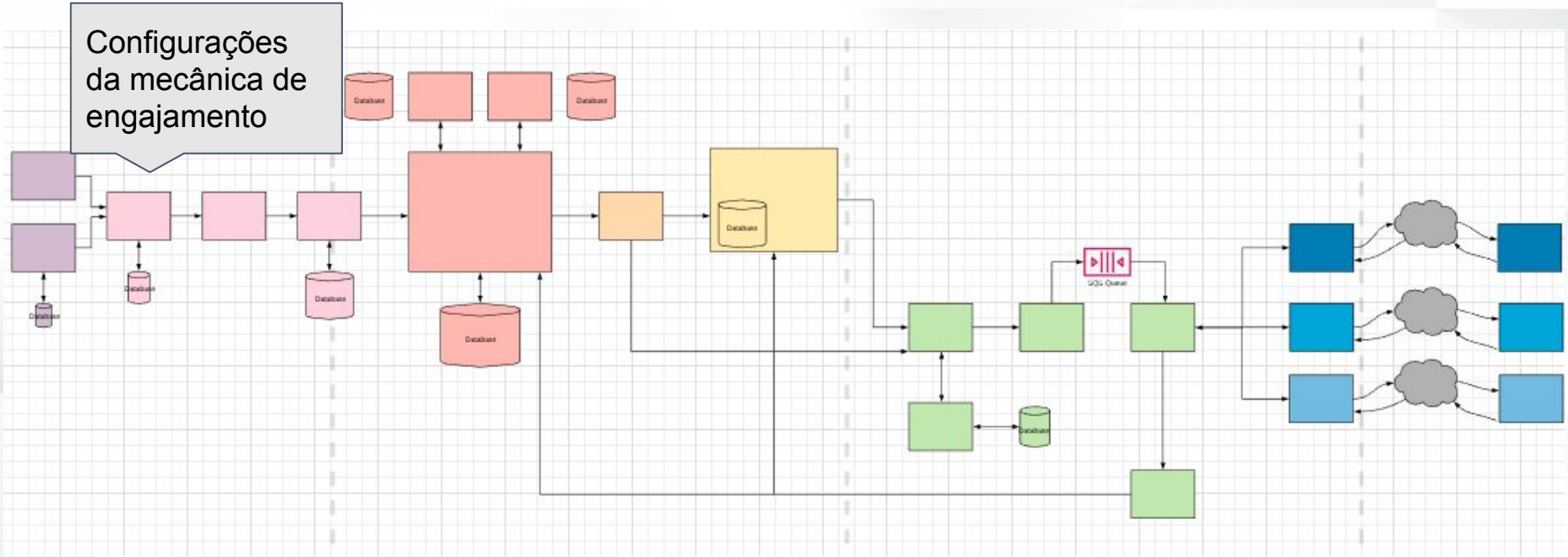


Contextualizando fluxo microsserviços

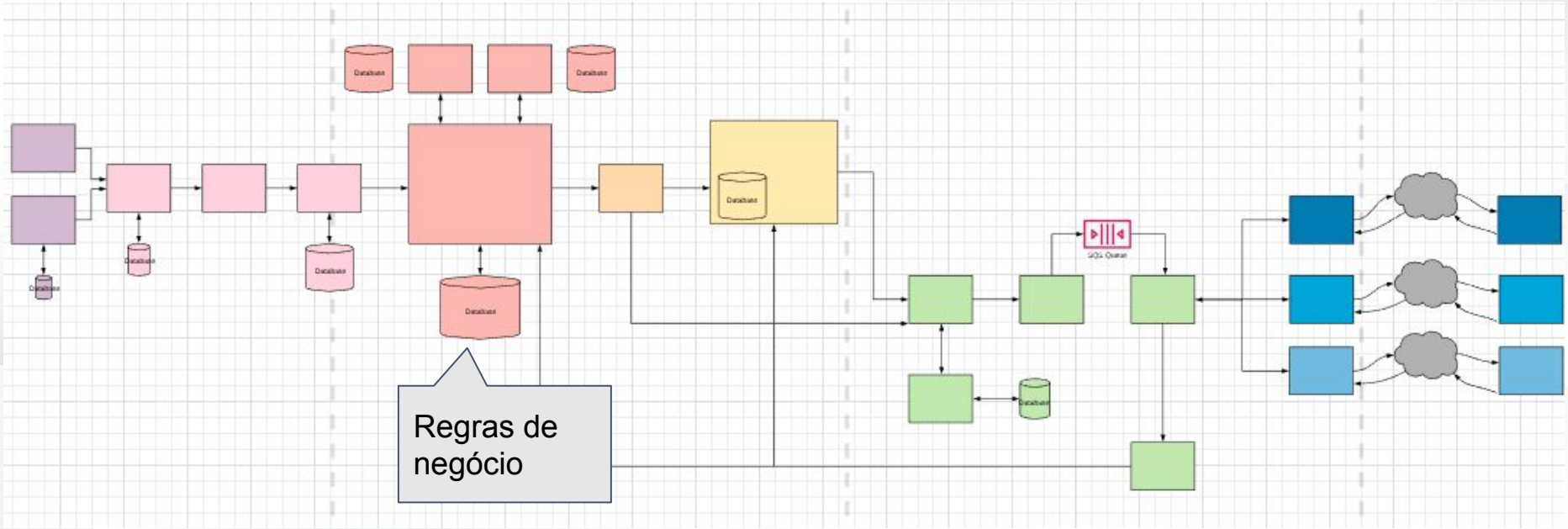
Entrada dos requests



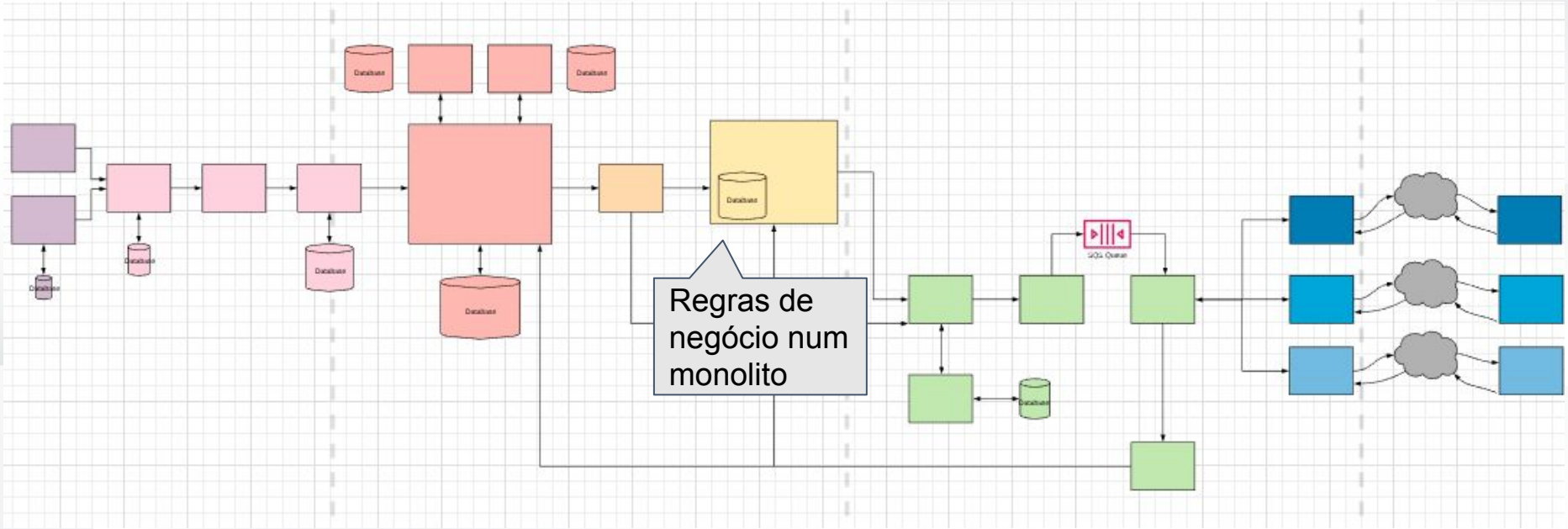
Contextualizando fluxo microsserviços



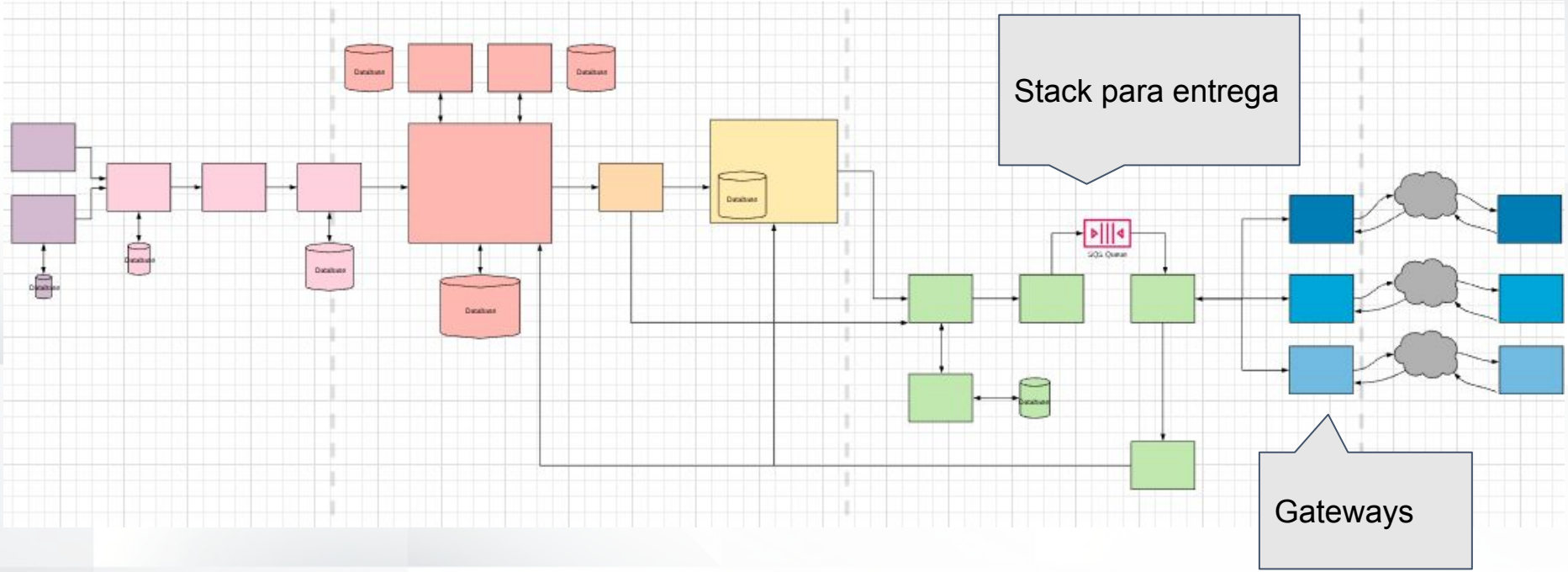
Contextualizando fluxo microsserviços



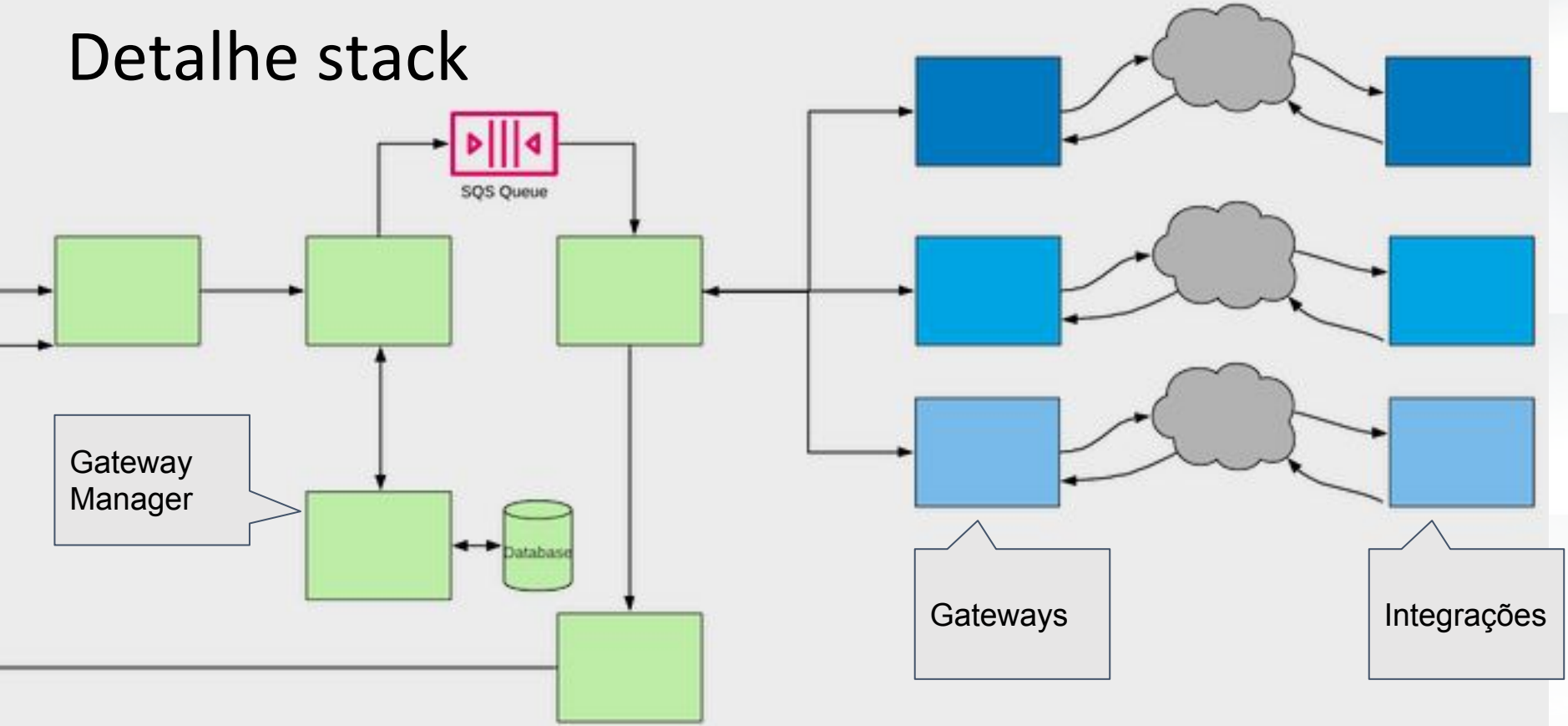
Contextualizando fluxo microsserviços



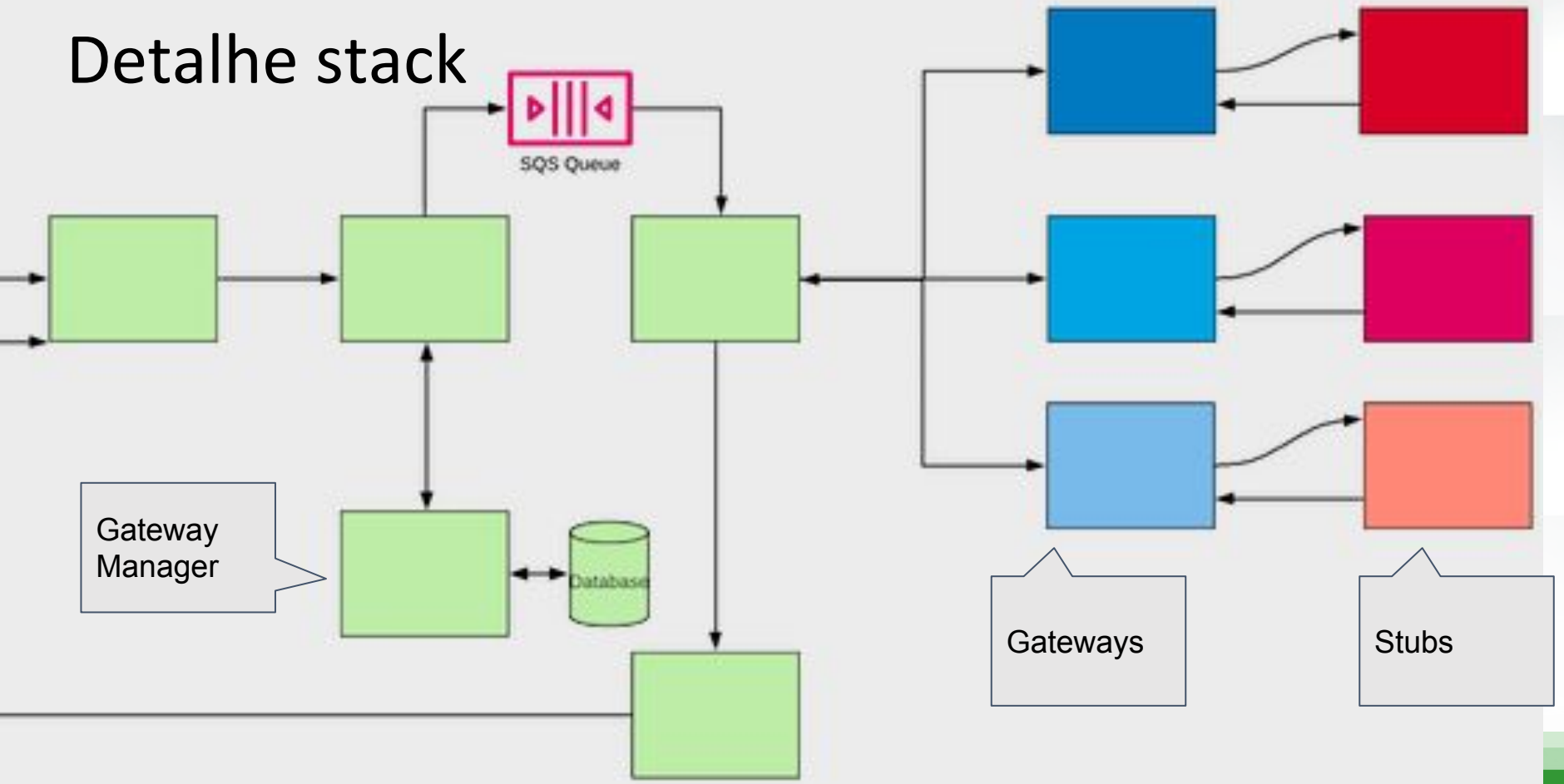
Contextualizando fluxo microsserviços



Detalhe stack

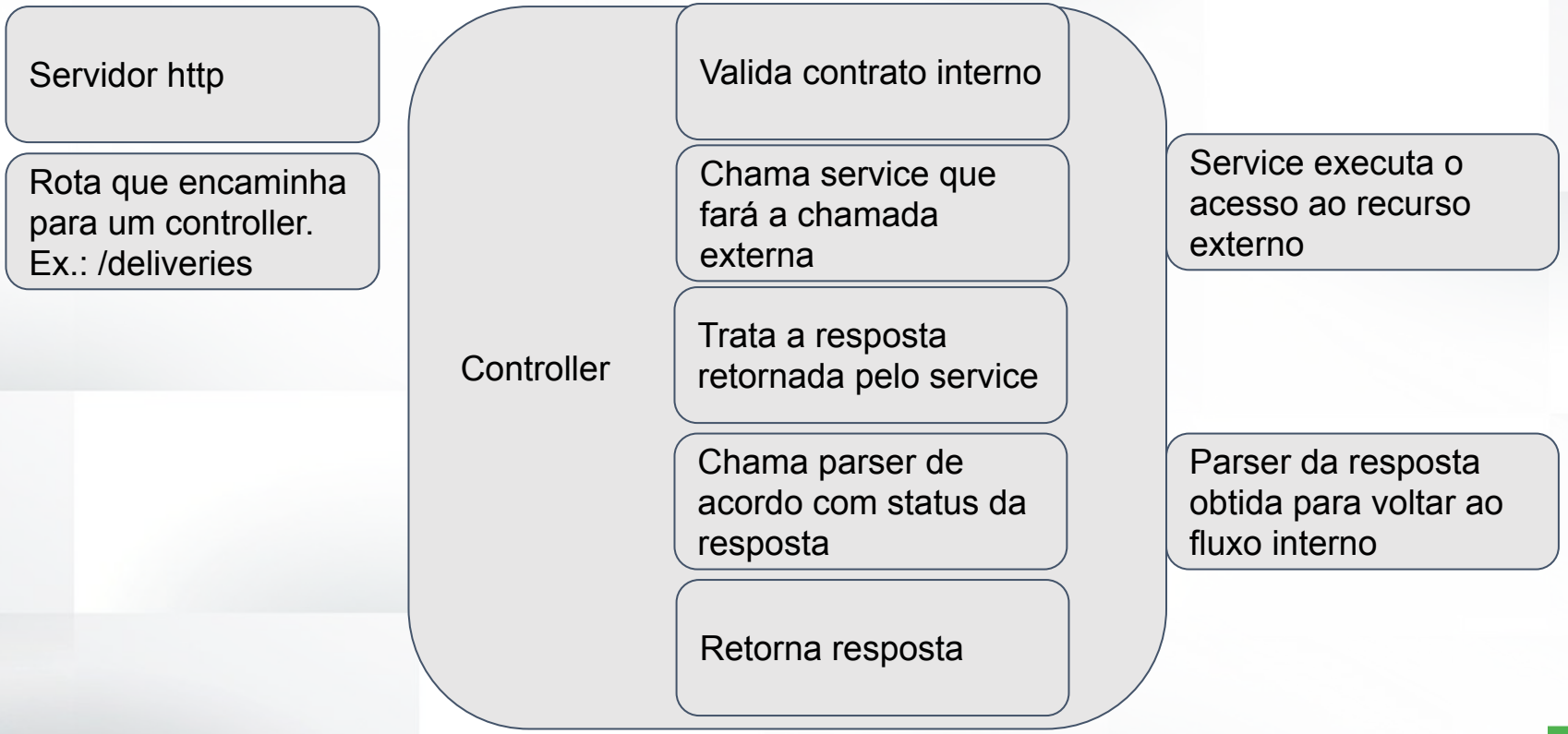


Detalhe stack



Anatomia Gateway

Anatomia de um Gateway



Anatomia de um Gateway - Estrutura de pastas

Anatomia de um Gateway - Controller parte 1

```
module.exports = (req, res) => {
  const { body } = req;
  const validationResult = validation.validate(body, 'gateway-entry-data');

  if (!validationResult.valid) {
    const validationError = validation.formatValidationError(validationResult);
    logger.error(`Bad Request on call this gateway. Body:${JSON.stringify(body)}.
    ErrorMessage: ${validationError}`);

    return res.status(utils.httpStatusCode.badRequest).send(validationError.toString());
  }
  const mobile = utils.formatPhone(body.delivery.deliveryRecipient.msisdn);
  const { paymentId } = body.delivery.data;
  const { product } = body.delivery;
  const { integrationUrl } = body.integrationConfiguration;
  const integrationToken = body.integrationConfiguration.authToken;
```

Validação utilizando tv4

Anatomia de um Gateway - Controller parte 2

```
postService(mobile, integrationUrl, integrationToken, paymentId, product, (err, resPostUser) => {
  if (err) {
    utils.analytics.registerEvent(utils.analytics.events.credit, null,
      utils.httpStatusCode.internalServerError);
    logger.error(`Response error from alliance: ${JSON.stringify(err)}`);

    return res.status(utils.httpStatusCode.badRequest).send(utils.doParse(err, body));
  }

  if (resPostUser.statusCode < 199 || resPostUser.statusCode > 299) {
    logger.warn(`Returned an error. Response statusCode [${resPostUser.statusCode}].
      Body received: ${JSON.stringify(resPostUser)}.`);

    return res.status(resPostUser.statusCode).send(utils.doParse(resPostUser, body));
  }

  logger.info(`Response success. Mobile: ${mobile}, Body response: ${JSON.stringify(resPostUser.body)}`);
  utils.analytics.registerEvent(utils.analytics.events.credit, resPostUser.timingPhases,
    resPostUser.statusCode);
  res.status(resPostUser.statusCode).send(utils.doParse(resPostUser, body));
});
```

Tratamento em caso de erro

Tratamento em caso de status de erro

Tratamento em caso de sucesso

Anatomia de um Gateway - Service

```
const { logger } = require('minutrade-logger');
const request = require('request');

module.exports = (mobile, integrationUrl, integrationToken, paymentId, product, callback) => {
  const options = {
    method: 'POST',
    url: integrationUrl,
    json: true,
    time: true,
    headers: {
      'Content-Type': 'application/json'
    },
    body: {
      token: integrationToken,
      campaign_id: parseInt(product, 10),
      phone: mobile.toString(),
      postId: paymentId
    }
  };
  logger.info(`Send Request IntegrationUrl: ${integrationUrl} - RequestOptions: ${JSON.stringify(options)}`);
  request(options, callback);
};
```

Implementa contrato externo

Anatomia de um Gateway - Parse da resposta

```
const doParse = (responseAlliance, entryBody) => {
  const transactionCode = entryBody.delivery.data.paymentId;
  const { statusCode } = responseAlliance;
  const responseData = {};
  if (statusCode === httpStatusCode.created && responseAlliance.body) {
    responseData.statusCode = config.get(
      `STATUS_MAPPING:${responseAlliance.statusCode}`
    );
    responseData.responseCode = responseAlliance.statusCode.toString();
    responseData.responseDescription = 'ProcessedOK';
    responseData.responseBody = JSON.stringify(responseAlliance);
    responseData.responseTransactionCode = responseAlliance.body.voucher;
    responseData.requestTransactionCode = transactionCode;
  } else {
    responseData.statusCode = config.get(
      'STATUS_MAPPING:UNEXPECTED_ERROR'
    );
    responseData.responseDescription = 'Error';
    responseData.responseCode = responseAlliance.statusCode
      ? responseAlliance.statusCode.toString()
      : null;
    responseData.responseBody = JSON.stringify(responseAlliance);
    responseData.requestTransactionCode = transactionCode;
  }

  return responseData;
};

module.exports = doParse;
```

Implementa a interpretação da resposta obtida

Pontos a observar

Testes automatizados

- Integração (nocks)
  - Fixtures de entrada para o sucesso e possíveis erros da integração
  - Testes utilizando as fixtures
  - Testes com as restrições do **T**iny **V**alidator (for v**4** JSON Schema).
- Unitário
  - Função que faz a interpretação do retorno
  - Funções úteis para aquele gateway

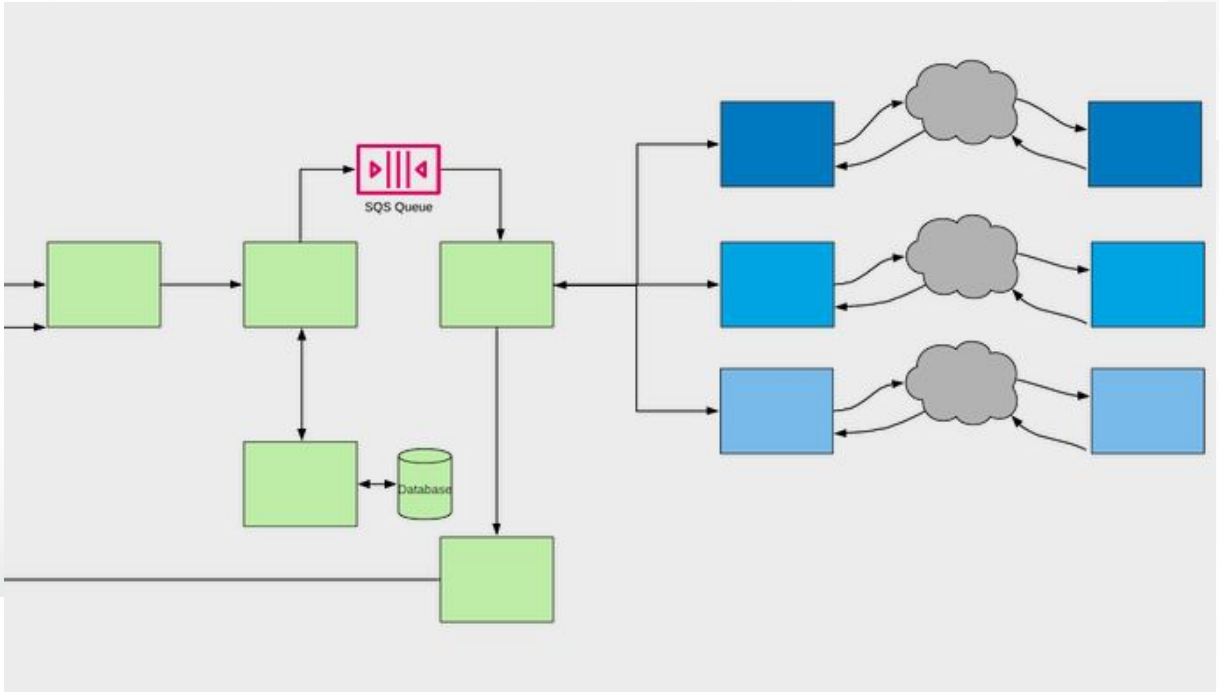
Lições aprendidas

Lições aprendidas

- Código duplicado
- Nomes genéricos
- Validação Json schema
- Stubs

Lições aprendidas

Diagramas para visualizar fluxo



Lições aprendidas

Destructuring

```
const { params: { id, user }, body: { expirationDate } } = req;
```

Lições aprendidas

Monitoramento

Lições aprendidas

Monitoramento

Espero que…

Obrigado pessoal!!!

Cassiano Raimar Silva

- Desenvolvedor na Minutrade
- Mestrando em Engenharia e Gestão de Processos e Sistemas - IETEC