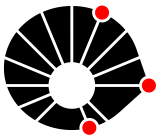

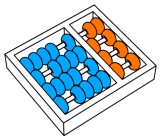

Projeto Arquitetural da Plataforma de Trabalho Decente

Waki, L. H. *França, B.*

Relatório Técnico - IC-PFG-21-18
Projeto Final de Graduação
2021 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Projeto Arquitetural da Plataforma de Trabalho Decente

Lucas Henrique Waki* Breno de França †

Resumo

Nos últimos anos, uma onda crescente de plataformas de trabalhos começou a surgir em meios digitais. Aplicativos que angariavam trabalhadores sem vínculo empregatício se tornaram populares e, com isso, a precarização de trabalhos de motoristas e entregadores, por exemplo, tornou-se uma discussão ativa na sociedade frente essas mudanças. Nesse contexto, surge a ideia da Plataforma do Trabalho Decente, com o intuito de oferecer como política pública serviços já realizados por trabalhadores autônomos, porém garantindo direitos trabalhistas mínimos como seguridade social. Este projeto final de graduação apresenta o projeto arquitetural para a plataforma em desenvolvimento, aplicando técnicas de Engenharia de Software. O projeto utilizou princípios *Domain-Driven Desgin* e modelos C4 para projetar um arquitetura escalável baseada em microsserviços que apoiará o desenvolvimento da plataforma.

1 Introdução

Nos últimos anos, uma onda crescente de plataformas de serviços começou a surgir em meios digitais. Aplicativos que angariam trabalhadores sem um contexto de vínculo empregatício se tornaram populares e, com isso, a precarização do trabalho de funções como motoristas e entregadores se tornou uma discussão ativa na sociedade frente essas mudanças. Esse tipo de situação acaba sendo vista e vivida também por outros trabalhadores autônomos, que têm dificuldade de ofertar sua força de trabalho e de estar de acordo com aspectos legais. Dessa forma, esses trabalhadores perdem a garantia de direitos trabalhistas como seguridade social, e deixam de cumprir com realidades que podem influenciar muito em sua vida - não tendo direito, por exemplo, a seguros relacionados a lesões e doenças, além de outras proteções vindas do Estado. Ainda, as exploração dos trabalhadores pelas plataformas tem submetido os mesmos a condições inadequadas de trabalho e jornadas indefinidas [9]. Na região metropolitana de Salvador, por exemplo, estima-se que existam cerca de 67 mil autônomos [6].

Tendo esse cenário em mente, este Projeto Final de Graduação inicia o projeto da arquitetura de uma plataforma denominada Plataforma do Trabalho Decente (PTD), que permitirá aos trabalhadores autônomos terem acesso a esses direitos e deveres, ao mesmo tempo expandindo a rede de clientes e ajudando a equilibrar o ecossistema econômico das atividades profissionais em um ambiente justo.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
†Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

A ideia da PTD é ser ofertada com atuação do Estado e cooperativas de trabalhadores autônomos, regulando de forma justa o nicho de fornecedores de serviços, garantindo direitos e deveres ao mesmo tempo que abrangendo o acesso a esses serviços oferecidos. Assim, tornando a plataforma uma verdadeira política pública de garantia de direitos trabalhistas.

Ao investigar técnicas de Engenharia de Software, mais especificamente tecnologias e soluções arquiteturais necessárias para o desenvolvimento da plataforma, esse trabalho apoia essa parcela da população que busca um sustento de forma autônoma - e ajudará no desenvolvimento de uma ferramenta que respeite os direitos e entregue seguridade para essa população.

O projeto desenvolveu em etapas uma arquitetura de software para a plataforma, em conjunto com outros trabalhos em andamento, focando em neste o desenvolvimento da arquiteturas e possibilidades de implementação. Esta interação, tem o foco primário em aplicar Engenharia de Software para levantar requisitos e estruturar a arquitetura que a plataforma se baseará. A escala e necessidades da plataforma envolvem desafios de mapeamento de requisitos, infraestrutura, segurança, arquitetura e escolhas de tecnologias não vistas durante o curso de graduação.

O restante do projeto está organizado da seguinte forma. A Seção 2 apresenta os métodos utilizados para o desenvolvimento do trabalho.

2 Métodos

O objetivo principal do projeto é analisar requisitos e projetar a arquitetura de uma plataforma para oferecimento de serviços autônomos com garantias de direitos trabalhistas, denominada Plataforma de Trabalho Decente. As subseções a seguir descrevem as etapas percorridas no desenvolvimento do trabalho.

2.1 Etapa 1: Embasamento teórico

Inicialmente, foi realizado um estudo sobre arquiteturas baseadas no estilo arquitetural de microsserviços, de forma a estabelecer as necessidades técnicas que envolveriam a realização de um projeto nessa arquitetura. A adoção do estilo arquitetural se justifica por uma arquitetura escalável e que permite o desenvolvimento por equipes paralelas, utilizando tecnologias distintas [8], dado que o projeto conta com colaboradores distribuídos geograficamente e que possuem metodologias de desenvolvimento distintas. Ainda, os serviços que compõem esse tipo de arquitetura utilizam interfaces leves de comunicação, usualmente *APIs REST*, por sua facilidade de *deploy* - de criar e erguer uma nova instância das API -, atualização mais simples e escalabilidade.

Para isso, foram utilizadas leituras como *A Pattern language for Microsservices* [1] para embasamento teórico. Ainda, foi estudado a notação C4 (*C4 Model*) para modelagem da arquitetura [2]. O piloto inicial da PTD é idealizado no estado da Bahia. Isso define a pela necessidade de uma estrutura que possa ser atualizada e que consiga suportar a carga de usuários - estimada em aproximadamente 15 mil autônomos (cerca de um quarto dos 67 mil autônomos[6] da região de Salvador).

2.2 Etapa 2: Entendendo o Problema

Foi realizada uma entrevista com o professor Prof. Dr. Vitor Araújo Filgueiras da Universidade Federal da Bahia, um dos idealizadores da plataforma. Nessa oportunidade, foram discutidas a realidade, a aceitação e utilização - tais como parcerias que o projeto poderia ter a longo prazo. Os idealizadores da plataforma elaboraram uma descrição do que seria o "Caminho Feliz" - conjunto de narrativas de utilização para o aplicativo e um cenário de completo sucesso. Esse documento foi analisado pelo autor desse projeto e pelo aluno Kauan Takeda (também colaborador do projeto), onde foram levantados quais os fluxos de atividades seriam necessários para a plataforma, resultando em uma especificação de funcionalidades por grandes áreas, que contém os domínios de negócio a serem cobertos pela plataforma completa.

2.3 Etapa 3: Desenvolvimento de arquitetura

Tanto o "Caminho Feliz"quanto a especificação dos domínios da PTD foram utilizados como base para a criação de um diagrama de domínios, com o objetivo de isolar as funcionalidades listadas e permitir uma visualização *top down* do que o sistema virá a ser. Essa visualização ainda não apresenta os aspectos técnicos pois representa uma modelagem *DDD* (*Domain-Driven Design*), abordagem descrita por Eric Evans[4], onde a ideia é aproximar os conhecimentos dos especialistas no domínio do desenvolvimento do software.

2.4 Etapa 4: Diagramação C4

Escolhemos utilizar a diagramação C4 [2] para representar a arquitetura final da plataforma. Uma vez que temos os fluxos e o Diagrama de Grandes Áreas, o trabalho é traduzir as informações que eles tem em funcionalidades, dividindo os contextos e separando os *Containers*. A grande vantagem dessa modelagem está na simplificação: ao utilizar o conceito de domínios junto com o conceito de separação do C4, podemos desenhar a própria arquitetura de micro serviços.

A diagramação funciona utilizando 4 *levels* para representar o sistema, de aumenta-se os detalhes conforme atinge-se uma profundidade maior nos contextos.

3 Arquitetura Proposta

A arquitetura proposta foi definida utilizando princípios de DDD e o estilo arquitetural em microsserviços. Com isso, um esboço inicial da divisão em contextos foi realizado, conforme apresentado na Figura 1.

Na figura, cada retângulo maior representa um contexto de atividades, que concentra atividades e processos similares. Dessa forma, podem especializar suas funções e discutir diretamente com especialistas em cada uma dessas áreas. Também é importante verificar as relações que as áreas tem uma com as outras, pois são essas relações que permitirão criar os micro serviços nas próximas etapas do trabalho;

Nas subseções a seguir, serão apresentados os diagramas seguindo o modelo C4 para detalhamento da arquitetura em três níveis distintos: contexto, contêiner e componentes.

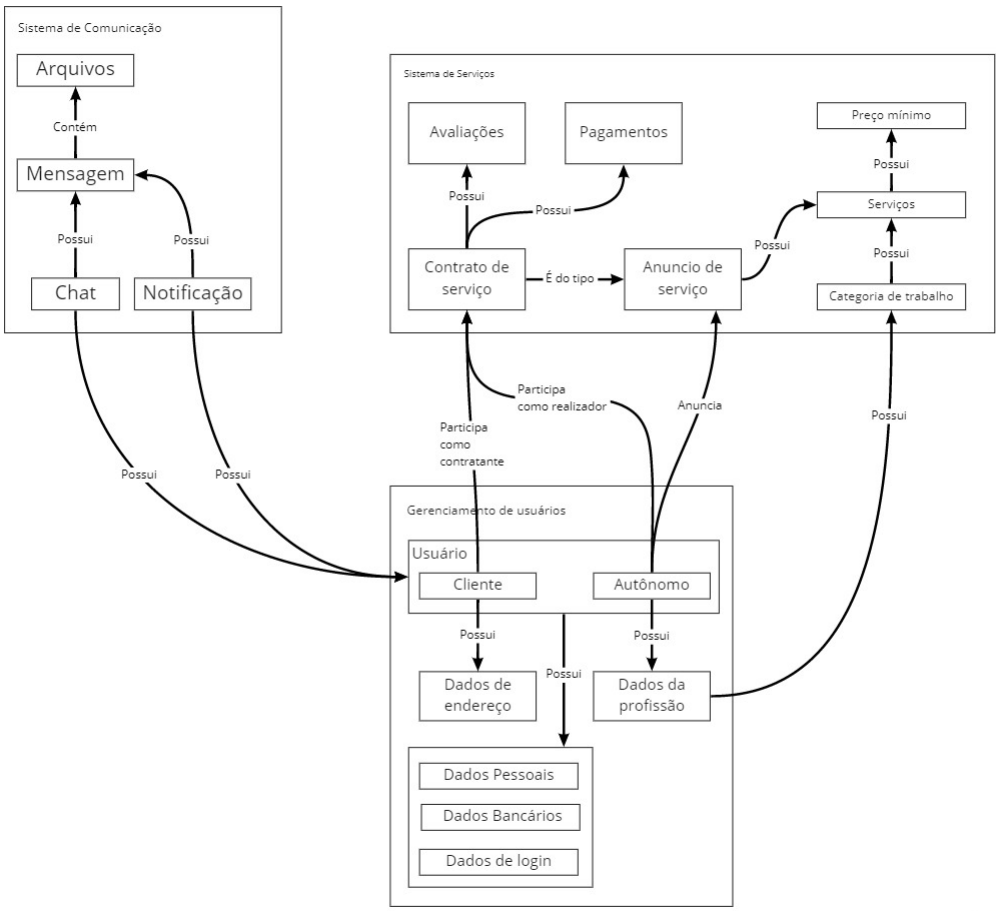

Figura 1: Diagrama de domínios, dividindo as grandes áreas da PTD

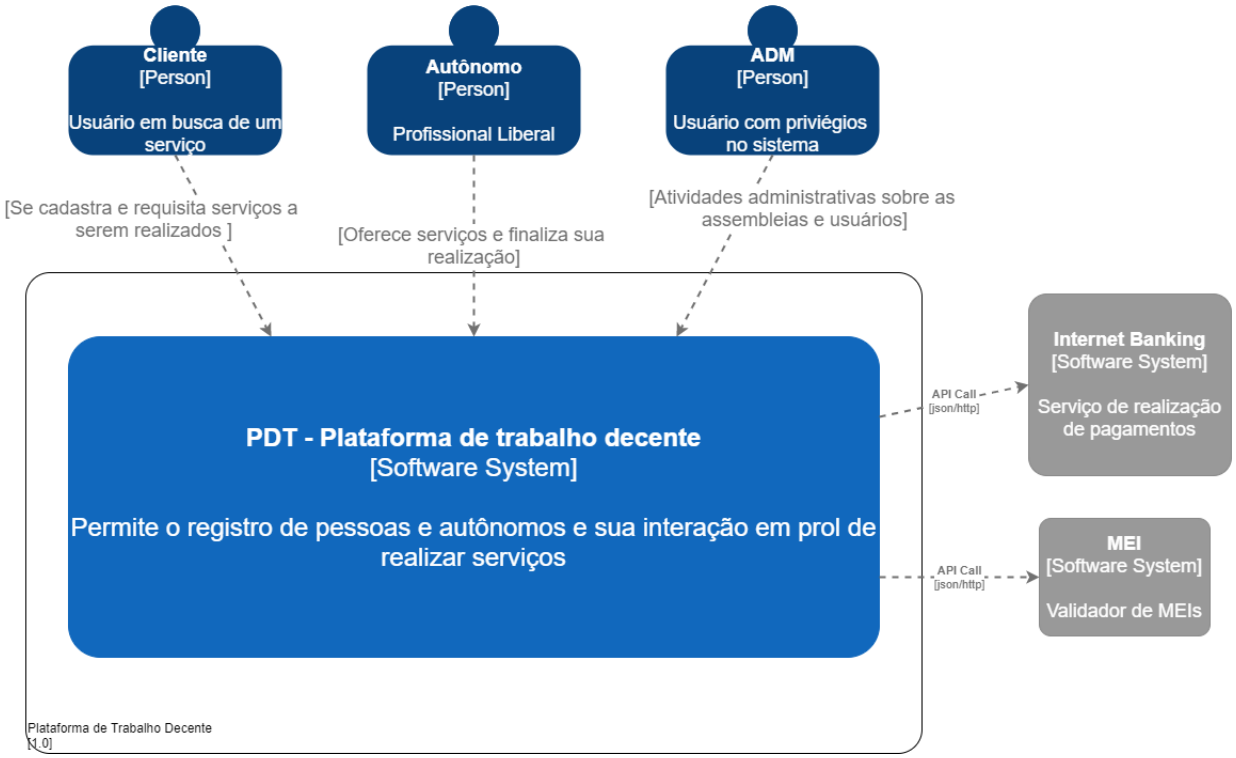

Figura 2: Diagrama C4 - Nível C1: Destaca-se a necessidade de interagir com sistemas externos e os papéis: Administrador do sistema, Autônomo e Cliente

3.1 Nível de Contexto

No nível C1 (Figura 2), representamos o sistema como um todo. É importante perceber quais são as interações que serão realizadas com o a plataforma por elementos externos (usuários e outros sistemas), onde percebemos que existe uma dependência com dois sistemas externos. O *internet Banking*, que fará a realização dos pagamentos e movimentações bancárias e o serviço MEI (Microempreendedor individual), que validará as informações dadas pelos autônomos. O último tem a função de validar as identidades de forma a evitar fraudes em cadastros e utilização de mão de obra ilegal.

Avaliando os usuários da plataforma, temos o Cliente - que vai pedir por um serviço, o Autônomo - que oferece o serviço, e o ADM - que é responsável por atividades administrativas do sistema. A interação entre esses papéis é o que tornam a plataforma funcional.

3.2 Nível de Contêineres

No nível C2, avaliamos a PDT - Plataforma de Trabalho Decente, e projetamos quais são os microsserviços que a constituirão. A preocupação com escalabilidade adicionou duas soluções interessantes: o padrão de projeto *API Gateway* e as filas *Kafka*, provendo serviços de mensageria.

A função do *API Gateway* é centralizar o fluxo de requisições que entra no ecossistema de APIs. Por ser o único ponto de entrada, ele pode controlar o número de instâncias de cada API tal qual seus endereços, escrever logs sobre o que está sendo requisitado, lidar com o trabalho de autenticação, além de centralizar a estrutura de segurança do sistema inteiro.

As filas *Kafka* são utilizadas para controlar o fluxo do informação que entrará em cada uma das APIs dos microsserviços. Essas implementam um estilo arquitetural conhecido como Publicador-Assinante (ou Invocação Implícita) [10]. As filas *Kafka* são tolerantes a falhas e adicionam robustez na solução arquitetural. É importante ressaltar que a utilização dessa estrutura faz com que a comunicação seja assíncrona entre o chamado que vai ser consumido e a API consumidora.

Os fluxos decididos na Figura 1 foram validados em reunião com os professores envolvidos, o que nos permitiu avançar com a estrutura de APIs: de serviço, de cadastro, de login, de pagamentos e de *chat*.

3.3 Nível de Componentes

Nos diagramas C4 em nível C3, foram definidos os módulos internos dos microsserviços. Com isso em mente, cada uma das APIs terá uma diagramação que busca refletir o que será o sistema final. Devido à mudanças de interpretação em cada um dos ciclos, e à mudanças de requisitos, esses módulos foram mais sensíveis à mudanças, de forma que sua implementação final pode necessitar de adaptações em relação ao projeto aqui apresentado.

3.3.1 API Cadastro

A API Cadastro (Figura 4) tem a função de centralizar todas as informações referentes aos autônomos, clientes, administradores e credenciais de autenticação. É API central de armazenamento, e também a responsável por realizar a verificação com o serviço MEI para validação de autônomos.

Tendo fluxos separados para o cadastro de cada um dos papéis, ela é composta por três controladores básicos responsáveis por essas informações: de Autônomo, de Cliente e de ADM, cada um responsável por dar acesso e inscrição no banco de dados sob o domínio o qual é responsável.

Por fim, o controlador de *login* administra os *tokens* e validações necessárias, além de ser acionado para a criação das contas e atribuição de permissão durante o cadastramento.

3.3.2 API Serviços

A API de serviços (Figura 5) é o "coração" da plataforma. É ela possibilita que os autônomos possam oferecer serviços, criar ordens de pagamentos e estipular valores nas atividades que realizarão. É essa API que mantém um registro de quais as atividades são lícitas e quais devem ser os preços mínimos para cada uma delas.

Percebe-se então os três controladores mais importantes da API: o de Anúncios, de Ordem de serviços e de Serviços.

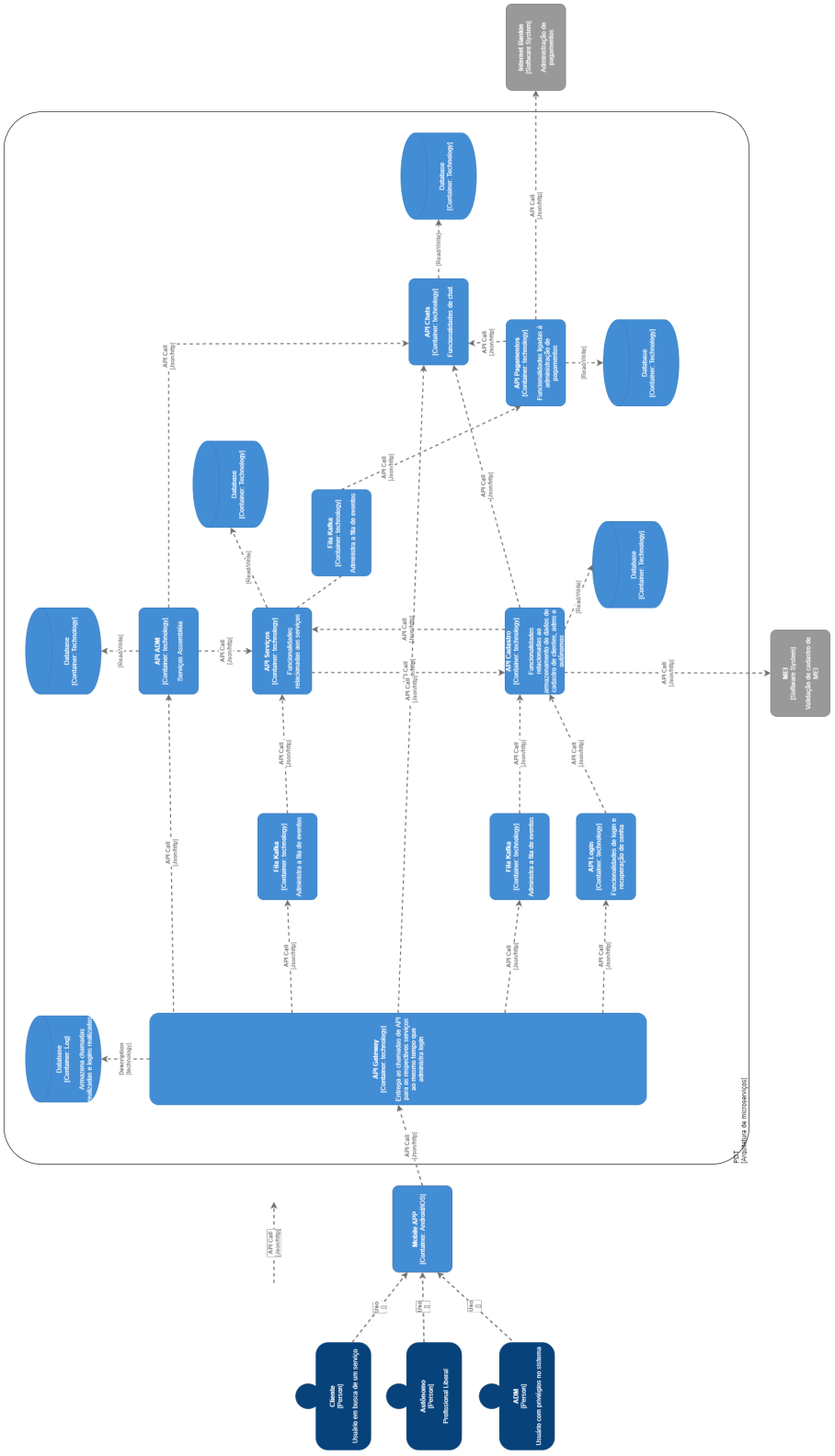

Figura 3: Diagrama C4 - Nível C2: Destaca-se a divisão dos serviços e em *contêineres* independentes, que são interligados diretamente ou por filas *kafka*; a imagem em maior resolução pode ser vista em: [5]

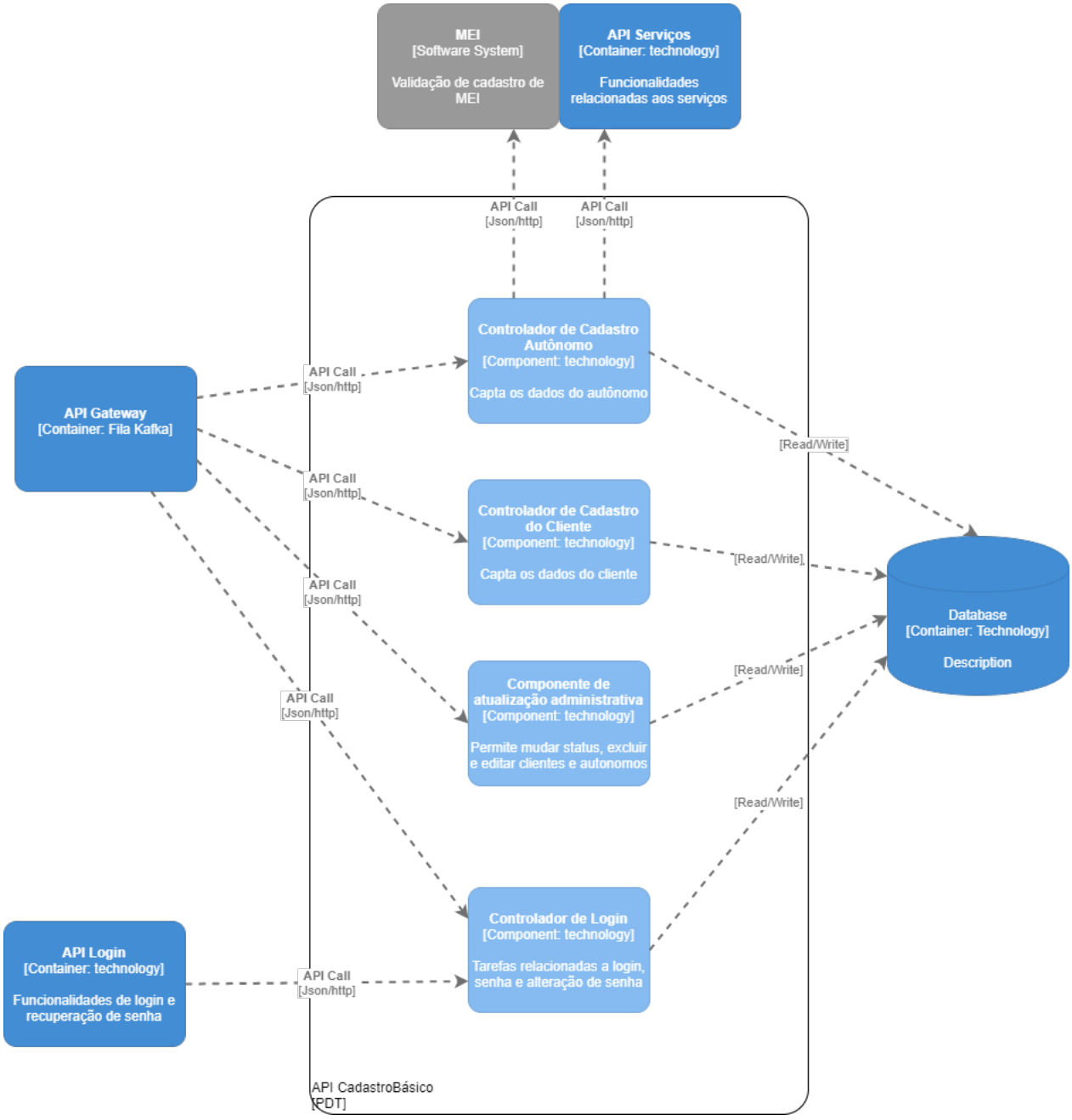

Figura 4: Diagrama C4 - Nível C3 - API Cadastro

- O controlador de Serviços administra a lista de serviços possíveis e os preços mínimos que devem ser aplicados. Também controla quais profissões tem acesso a quais serviços.
- O controlador de Anúncios permite que o autônomo cadastre os serviços que pode realizar de acordo com o seu cadastro
- O controlador de Ordem de Serviços atua quando um serviço for confirmado e é necessária a criação e registro de uma ordem de pagamento. Serviços de verificação e de confirmação do andamento da atividade também ficarão centralizados nesse controlador, de forma que a API de pagamentos deverá ser acessada para permitir o registro da ordem e as devidas obrigações de leis que isso implica;

3.3.3 API Pagamentos

A API de pagamentos (Figura 6) é a ferramenta que vai gerenciar o trânsito de informações financeiras dentro do aplicativo. É responsável por entrar em contato com o serviço de *internet banking* e administrar os estados das transações financeiras pelo aplicativo.

O controlador de ordem de pagamento ficará responsável por checar o estado das transações no *internet banking*. Ainda, o controlador de impostos deve fazer a separação e cálculos referentes à coleta de impostos. Não foi decidido sobre qual o modelo de pagamento estipulado, tendo sido estudado as seguintes possibilidades:

- A plataforma recebe o pagamento, separa os deveres e repassa os direitos ao autônomo.
- O autônomo recebe o pagamento e tem que pagar os direitos por sua conta.
- A plataforma gera dois pagamentos: os direitos e os deveres, separadamente.

A avaliação sobre a forma de aplicação deverá ser feita com base no *internet banking* a ser utilizado, assim como averiguação com a parte legal sobre o funcionamento desse tipo de pagamento. Tais informações serão implementadas no controlador de impostos.

O controlador de ordem de pagamentos também terá autonomia para enviar mensagens via *chat* usando a API *Chats* para notificar o autônomo/cliente sobre o estado de ordens de pagamentos.

3.3.4 API Chat

A API Chat (Figura **??**) é responsável por permitir a comunicação entre os usuários. A função dela é armazenar as informações relacionadas à mensagens e ser acionada sempre que for necessário fazer trocas de mensagens entre Cliente, Autônomo e ADM.

O controlador de *Flood*/notificação é um controlador que funciona disparando mensagens para grupos, que podem ser lidas como notificação ou mensagens em chat. O controlador de Conversas é responsável por administrar as mensagens que estão sendo trocadas entre os usuários. Finalmente, o controlador *newChat* é acionado pela API de Serviços, para permitir que haja conversa entre os envolvidos em um serviço.

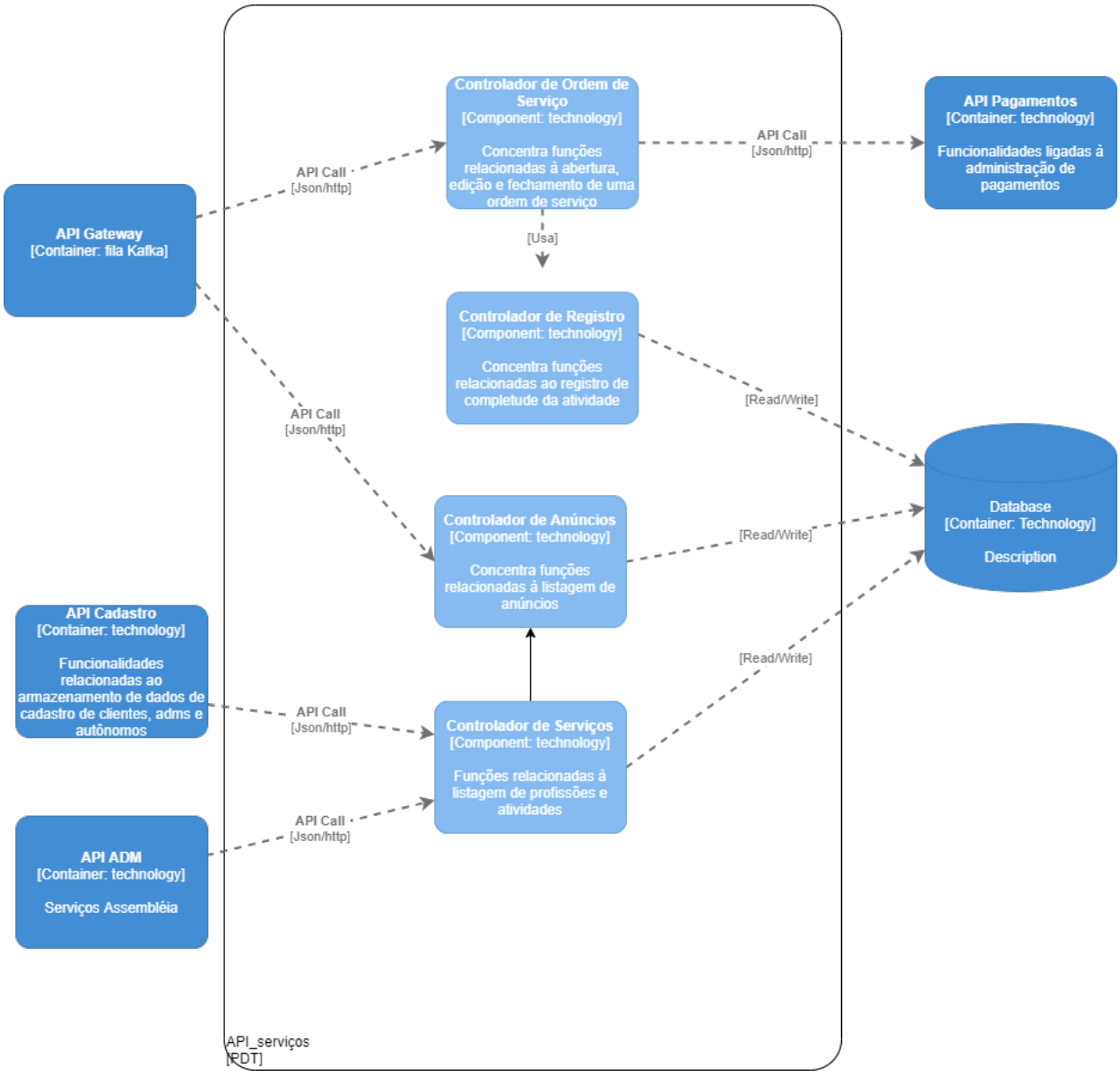

Figura 5: Diagrama C4 - Nível C3 - API Serviços

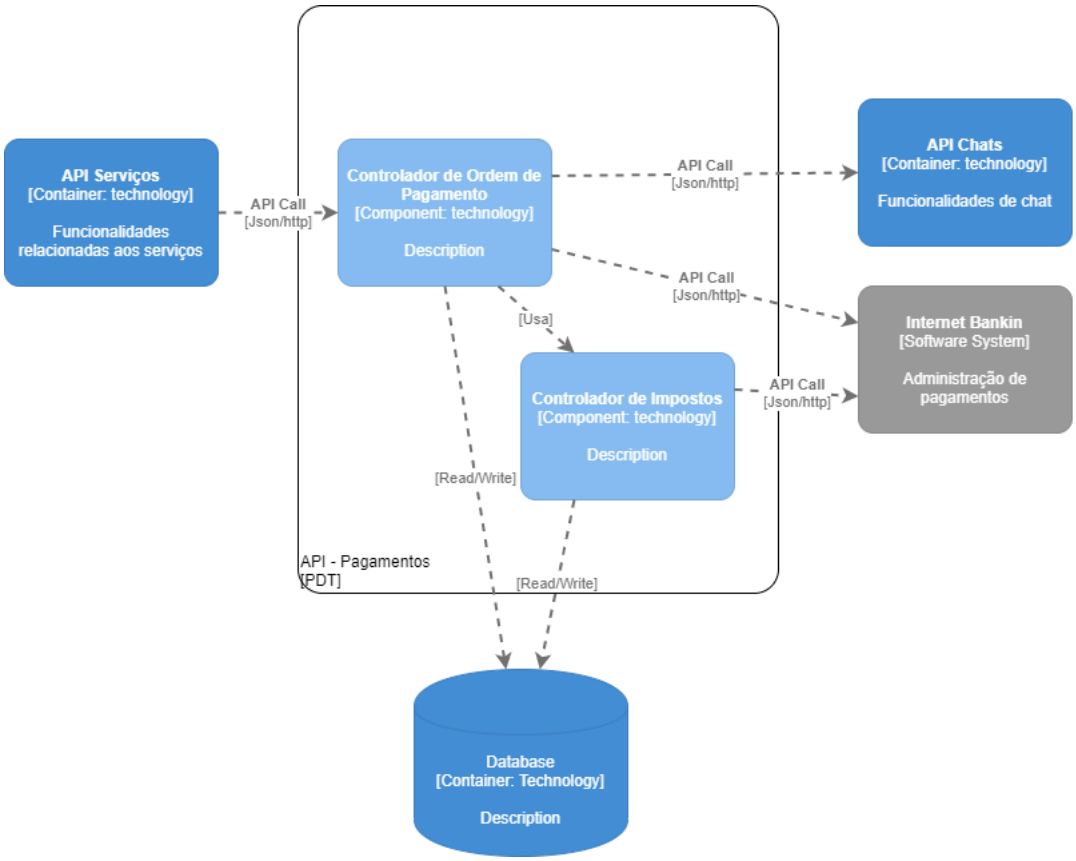

Figura 6: Diagrama C4 - Nível C3 - API Pagamentos

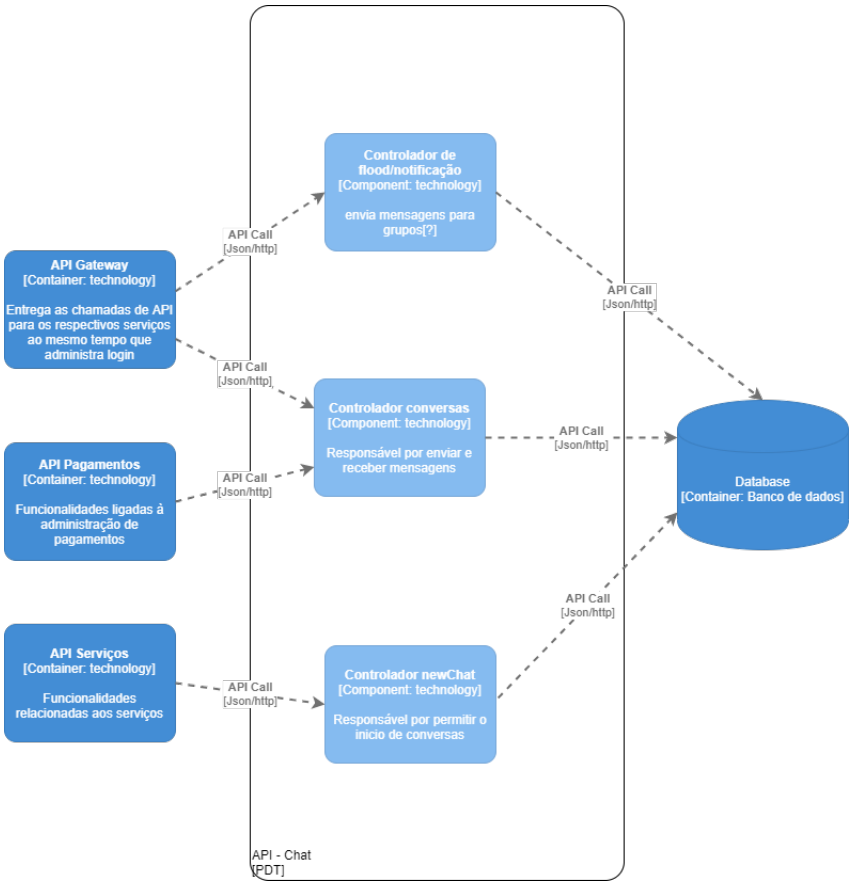

Figura 7: Diagrama C4 - lvl 3 - API Chat

4 Considerações Finais

Ao fim do período útil para realização do projeto, conseguimos finalizar os três primeiros níveis do Diagrama C4 e o a lista de fluxos da plataforma[7], artefatos importantes para a constituição de um software de alta complexidade. Como ainda existe um potencial de mudança na arquitetura, não foi esperado que o PFG realizasse a descrição dos métodos internos de cada controlador - o nível C4, deixando a liberdade de implementação a encargo da próxima equipe de continuar com o trabalho.

Concluímos que a maior parte do tempo da atividade foi investida com o levantamento de requisitos, preparação e entendimento de técnicas de Engenharia de Software. Levantar os requisitos, estabelecer os fluxos e desenvolver uma arquitetura que seja aprovada consumiram tempo, com duas versões da arquiteturas negadas, por exemplo. Entretanto, tal investimento se mostra um gasto necessário para pavimentar decisões com menos equívocos e menos custos durante a implementação.

Foram realizados teste de levantamento de APIs em *Python*, assim como de filas *kafka*, mas sem tempo hábil para implementar um protótipo da arquitetura. Tarefas a serem realizadas em atividades futuras do projeto.

5 Agradecimentos

Agradecemos à equipe que apoiou e acompanhou o desenvolvimento das atividades, citando o Prof. Dr. Vitor Araújo Filgueiras, o doutorando em Economia Leonardo Moura Lima Calmon de Siqueira e a graduanda em Ciências Sociais Victória Victor Vilas Boas da Silva.

Referências

[1] A Pattern language for Microsservices. **Patter Language for Microsservices**,2021. Disponível em : <https://microservices.io/patterns/index.html >acesso em: 11 de Julho de 2021

[2] C4 Model. **C4 Model**,2021. Disponível em <https://c4model.com/ >acesso em: 11 de Julho de 2021

[3] REDHAT **O que é uma API REST?**,2021. Disponível em <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api >acesso em: 11 de Julho de 2021

[4] E. EVANS, <Domain-Driven Design: Atacando as complexidades no coração do *software* >2016

[5] L. H. Waki <Diagrama lvl 2 - PTD,2021 Disponível em <https://im.ge/i/2lTsW >acesso em: 13 de Julho de 2021

[6] SEI - Superintendência de Estudos Econômicos e Sociais da Bahia **SEI divulga dados sobre mercado de trabalho na região metropolitana de Salvador em Fevereiro**,2021. Disponível em <https://www.sei.ba.gov.br/index.php?option=com_content&view=article&id=2700:sei-divulga-dados-sobre-mercado-de-trabalho-na-regiao-metropolitana-de-salvador-em-fevereiro&catid=10&Itemid=565 >acesso em: 11 de Julho de 2021

[7] K. Takeda e L. H. Waki **Funcionalidade por Grandes Áreas**,2021. Disponível em <https://docs.google.com/document/d/1GkCeJBXc4DYzixeyKRvF8QfeqhXSYyrewEIM75CiqGg/edit?usp=sharing >acesso em: 11 de Julho de 2021.

[8] J. Lewis, e M. Fowler. Microservices: a definition of this new architectural term. MartinFowler.com. 2014.

[9] Uol **GetNinjas levanta R$ 550 mi na Bolsa em meio a críticas trabalhistas**,2021. <https://economia.uol.com.br/colunas/carlos-juliano-barros/2021/05/18/get-ninjas-ipo-criticas-trabalhistas.htm >acesso em: 11 de Julho de 2021

[10] M. Shaw, and D. Garlan. Software architecture: perspectives on an emerging discipline (Vol. 1, p. 12). Englewood Cliffs: prentice Hall. 1996.