UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Liferay Portal Upgrade: definição de um processo eficiente para upgrade de clientes em versões legadas

---

Relatório Técnico relativo ao Trabalho de Conclusão Curso do Bacharelado em Sistemas de Informação na modalidade Empresa

---

Aluno
Nícolas Moura do Canto Ferreira

Orientador
Victor Wanderley Costa de Medeiros
Departamento de Estatística e Informática

15 de outubro de 2022

NÍCOLAS MOURA DO CANTO FERREIRA

LIFERAY PORTAL UPGRADE: DEFINIÇÃO DE UM PROCESSO EFICIENTE PARA UPGRADE DE CLIENTES EM VERSÕES LEGADAS

Relatório Técnico relativo ao Trabalho de Conclusão Curso do Bacharelado em Sistemas de Informação na modalidade Empresa apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado em: 11 de Outubro de 2022.

BANCA EXAMINADORA

Victor Wanderley Costa de Medeiros (Orientador)
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

Rodrigo Gabriel Ferreira Soares
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

Resumo

A *Liferay* propõe que o setor de engenharia intervenha no processo de migração do *Liferay Portal*, das versões mais antigas para a mais atual disponível no mercado. Processo anteriormente feito pelo departamento de consultoria, o *upgrade* é um trabalho que demanda tempo da equipe e acaba sendo repassado em valores altos para os clientes que optam por permanecer em versões desatualizadas. Para atingir suas metas, o time faz uso de tecnologias internas do produto *Liferay Portal DXP* com arquitetura separada em três camadas de *User Interface*: *core*, serviços e opcional. Elas consistem em aplicações *DXP web* para adicionar portais, sites, páginas, *widgets* e conteúdos para os diversos produtos. Além de trabalhar com *scripts* de *upgrade* de versões para tabelas nos bancos de dados dos clientes e fazer automações para esse processo na ferramenta interna de formatação de código da empresa. A engenharia apresenta assim uma nova proposta para processos de *upgrades* de clientes visando agilizar o trabalho do time de consultoria. Além de contribuir com documentação interna e automação para migrações futuras. Este trabalho visa discorrer sobre os eventos que circundam este projeto desde sua concepção, à organização do fluxo de trabalho, além da influência da graduação em sistemas de informação pela UFRPE no atual mercado de trabalho.

Sumário

1 Introdução

A *Liferay* é uma multinacional, focada em soluções digitais personalizadas, que estão inclusas em seu produto *Liferay Portal.* Este produto é uma solução de portal corporativo de código aberto e a *Liferay* até pouco tempo baseava completamente seus negócios na oferta de plataformas como serviços. Parte ativa na mudança de contexto de negócios para também ofertar programas como serviço está o escritório localizado em localizado em Recife-PE. É onde está localizado um dos maiores setores de engenharia de *software* da empresa, e o time *Hotel* é seu integrante. Este time surgiu de acordo com a necessidade de fazer *upgrades* de clientes de versões mais antigas do *Liferay Portal* para a sua versão mais atual, visto que manter um cliente em versões mais antigas não é ideal para a empresa. Uma vez que, o custo de manutenção é alto e fazer *upgrades* individuais para cada cliente junto ao time de consultoria, *Global Services (GS)*, acaba sendo custoso para o cliente que irá pagar por cada hora trabalhada dos consultores.

Neste contexto, se iniciou o projeto de *Customer's Upgrade* no *Hotel*, no qual ao invés de um time de *GS* fazer a migração total de um cliente, este time da engenharia fica responsável por parte dela. A ideia é migrar parte do código acelerando o processo de *upgrade* e compreender possíveis falhas de código que necessitem ajustes para mais de um cliente, implementando a mesma solução o que não seria possível através de consultoria personalizada, além de implementar um processo de automação. Isso pode então melhorar o aproveitamento de tempo dos consultores permitindo que eles foquem cada vez mais em personalizar estritamente o que é necessário para cada cliente, fazendo com que os projetos se tornem mais rápidos e portanto menos custosos para os clientes, o que acarretará em uma atração maior de clientes novos além de manter os antigos. Fazendo assim com que a empresa se torne cada vez mais competitiva no mercado.

Este trabalho está dividido da seguinte forma: a Seção 2 traz um breve histórico da empresa e sobre a influencia do aprendizado das tecnologias *Liferay* na atuação no *Hotel team*. A Seção 3 trata das problemática e a solução proposta, juntamente com tecnologias utilizadas e as contribuições. A Seção 4 apresenta as dificuldades encontradas no decorrer do projeto, por fim, a Seção 5 fala sobre os impactos da formação em bacharel em sistemas de informação no trabalho.

2 A empresa e sua atuação

A *Liferay* é uma multinacional espalhada por 19 países, com 16 anos de experiência no mercado de tecnologia e atualmente conta com mais de 1000 colaboradores no sistema presencial e *home office*. A empresa, idealizada em 2000, começou por um grupo de jovens engenheiros de *software*, uma dona de casa e uma futuro advogado cercados por móveis doados por uma igreja local à qual fazia parte o idealizador do projeto, Brian Chan [LIFERAY 2022d]. Nas palavras de Brian a "'Liferay fornece vários módulos predefinidos que permitem a criação de sites sócio-colaborativos"' [CHAN 2014]. Além de ter como um de seus principais objetivos criar tecnologia útil que beneficia quem dela possa usufruir e construir uma organização que ajude outras pessoas a atingirem seu pleno potencial. Essa é inclusive parte das missões da organização: ajudar as pessoas a atingir seu máximo potencial para servir aos outros.

São visões assim que regem o pensamento crítico da empresa, desde o momento que ela passou de um pedido pessoal de integrantes da igreja àqueles que a frequentavam e reuniões com clientes em mesas de piquenique para escritórios ao redor do mundo. Além do código aberto também se construiu uma plataforma empresarial digital líder, e o pequeno grupo de jovens da igreja passou a ser reconhecido como líderes de negócios dedicados que ainda participam ativamente nas tomadas de decisões da organização. Por exemplo, Brian ainda é o arquiteto de software chefe aprovando *pull requests (PR)* e constantemente ensinando outros desenvolvedores o que ele conseguiu juntar em sua experiência de anos na área e em seu contato com pessoas de tantos *backgrounds* diferentes. Eu tive a oportunidade de trabalhar em uma equipe diretamente com ele tendo várias oportunidades de aprendizado, não somente relacionado ao código, mas também à negócios e as relações com os clientes. Em sua entrevista para o DevCon [CHAN 2014], o co-fundador da empresa afirma que apesar de muitas empresas de *software* crescerem e terem como objetivo serem vendidas, esse não é o caso da *Liferay*. Seu intuito é usá-la para tornar suas vidas mais felizes e espalhar o DNA único da empresa através do mundo.

A *Liferay* atua no setor terciário da economia, considerando que suas atividades estão relacionadas à prestação de serviços relativos "'ao tempo, lugar, forma e benefícios psicológicos."' [KON 1996], por ser assim serviço de informática que beneficia o consumidor ao pensar em soluções que lhes forneça informações de forma mais útil à sua realidade. Alguns casos de sucesso divulgados pela própria empresa são os do banco italiano *BPER Banca* e a *Airbus Helicopters* [LIFERAY 2022a]. A *Liferay* ajudou a inovar o site da *BPER Banca* renovando sua imagem, mas permanecendo fiel à marca, usando tecnologia inovadora como segmentação de público alvo, animações e vídeos, além de ter conteúdo personalizado e navegação otimizada. Já para a *Airbus Helicopters*, que recorreu à *Liferay* para redesenhar seu portal do cliente, foi desenvolvida uma solução mais moderna adaptada à jornada de seus clientes de aproximadamente 24000 usuários melhorando o tempo de comercialização e otimizando o acesso tanto a informações quanto a serviços.

No que diz respeito à estrutura organizacional, Brian Chan explica que há oito anos era possível manter uma relação pessoal com alguns colaboradores, visto que viveram em sua casa nos Estados Unidos por meses enquanto o próprio Brian os treinava como escrever código da maneira que a *Liferay* desenvolve além de aprender sobre os valores e missões da empresa [CHAN 2014]. Foi exatamente o caso de Bruno Farache, que em 2010 começou a operação da *Liferay Latam* com *headquarters* baseado em Recife, uma operação focada em engenharia de *software* e de construção de soluções tecnológicas que se transformou em uma unidade comercial [LIFERAY 2020]. Como um de seus funcionários menciona em um vídeo institucional [LIFERAY 2017], é possível que qualquer colaborador trabalhe ao lado de um vice-presidente ou mesmo de um dos fundadores da empresa, não há escritórios com portas fechadas, apenas salas designadas para reuniões de uso comum a todos os funcionários. É, portanto, possível, comum e extremamente aconselhável que desenvolvedores inexperientes recorram aos mais seniores, inclusive ao próprio Brian, sempre que já esgotarem as suas próprias capacidades de solucionar um problema. Isso faz com que se opere com um ar mais informal e que as interações entre as pessoas não estejam puramente podadas pelas amarras da hierarquização.

O que Brian Chan [CHAN 2014] continua explicando é que ao fazer parte do treinamento dos colaboradores ele é capaz de perceber as nuances das habilidades pessoais de cada um e alocar esses recursos de acordo com a necessidade da empresa e a aptidão pessoal de cada um, pois para

ele todos são construídos de forma diferente e todos têm habilidades únicas. Essa é uma visão que se espera dos gestores, que eles possam guiar e ajudar os mais *juniors* a encontrar seu caminho dentro do setor de tecnologia. Entretanto, essa é de certa forma uma organização que vem sendo mudada nos últimos tempos, cada vez mais sendo reconhecida e tomando seu espaço no mercado tecnológico, considerada como uma das empresas desafiadoras pelo Quadrante Mágico para DXP do Gartner [LIFERAY 2021], a empresa vem crescendo em número de funcionários e isso faz com que a estrutura organizacional anteriormente usada não seja mais tão prática no dia a dia. Todos os envolvidos na organização vêm em um esforço diário e incansável para reestruturar a empresa de uma forma que sua operação se torne mais confortável para todos e ainda mais confiável para os clientes sem que seus valores essenciais sejam perdidos, sendo eles: produzir com excelência, liderar ao servir, valorizar as pessoas, crescer e melhorar, e continuar *nerdy* [LIFERAY 2022d].

Ao se apresentar com seu bordão *"Uma plataforma. Infinitas soluções."* [LIFERAY 2022e] a empresa oferece para seus clientes uma solução de portais empresariais com diversas ferramentas integradas para montar uma solução perfeitamente adaptada ao que o cliente deseja. Sendo esse o *Liferay Portal*, seu principal produto, que pode ser definido como um *framework* para construção de portais no qual é possível compor e modelar sua solução com suporte a intranet, extranet, controle de usuários, *roles*, permissões, fluxo de aprovação, tudo integrado dentro de uma mesma solução. Por ser *Open Source*, fornece documentação gratuita sobre seus produtos e serviço profissional pago aos utilizadores do seu *software*, sendo facilmente integrável à outras tecnologias. O cliente pode simplesmente baixar o código fonte do *Liferay Portal* e começar a modelar sua solução a qualquer momento, customizar da maneira que desejar e realizar as integrações que achar necessárias, caso já não fossem possíveis com código nativo do próprio Portal, versão conhecida como *Community Edition*.

A *Liferay* tinha como oferta padrão seu serviço de suporte e consultoria pagos para auxiliar a desenvolver utilizando *Liferay*, também era possível adquirir serviços exclusivos do *Liferay*, essa sendo a versão *Liferay Portal DXP*. Anteriormente, O *Liferay Portal DXP*, era vendido exclusivamente como um pacote, bem customizável, onde a cada necessidade do cliente era adicionada a possibilidade de adicionar uma *feature* existente customizável com o suporte necessário a ela, serviços como o *Analytics Cloud*, entre outros.

A arquitetura do *DXP/Portal* tem três partes principais: o *Core*, *Bootstraps DXP* e seus *frameworks*. O *Core* provê a *runtime environment* para gerenciar os *Services, UI components*, e customizações no Portal; *Services*: Expõe as *features DXP* e as *features* personalizadas através de *APIs Java* e *APIs web; UI*, a *UI* opcional da aplicação web para adicionar portais, sites, páginas, *widgets*, e conteúdo. É possível utilizar a *UI* e os *Services* em conjunto ou se concentrar somente na utilização de *Services* através de *APIs REST* ou de *APIs GraphQL*.

A Liferay tem um setor de desenvolvedores que é chamado de ”engenharia”que são onde ficam os desenvolvedores responsáveis pela adição de novas *features*, correção de *bugs*, controle de qualidade dos produtos da empresa. Esse setor é divido no que são chamados de ”times”dentro da empresa, cada um responsável pelo desenvolvimento e manutenção de um ou mais produtos da empresa. Trabalho na Liferay desde Janeiro de 2021, minha entrada se deu através de seu programa de estagiários, ”Programa de Imersão”, a proposta é que turmas de 24 estagiários entrem na *Liferay*, passem por um programa de treinamento escrito pelo próprio Brian, o *Basic Training*. Esse treinamento varia de duração para cada colaborador que entra na empresa, mas o seu tempo mínimo

é de duas semanas e passeia por tópicos importantes para a empresa como *Fedora, Git, React, Java, Database*, entre outros. Como o nome já diz, o programa de imersão objetiva passar uma imersão por tecnologias e projetos da empresa, e logo após um breve treinamento os estagiários já possam ser integrados em times, aprendendo com demandas reais. Fui integrante da T1, primeira turma de estagiários com um número tão grande de participantes de uma vez só, e assim que terminamos o treinamento fomos alocados para escrever testes automatizados. Testes estes que eram escritos em Poshi, linguagem de automação de testes baseada em Selenium criada pela própria empresa visando a automação de testes para seus produtos.

Após minha atuação na automação de testes, fui chamado para fazer parte do início de uma ideia de Brian Chan, ideia essa que viria a se tornar o *Solutions Team* dentro da empresa, que é atualmente o maior time dentro da engenharia com uma média de 32 desenvolvedores. Durante minha participação neste time passei para o cargo de *Associate Software Engineer 1*, lá fiz parte da criação e desenvolvimento direto de produtos como o *Click to chat, DigitalSignature, Site Initializer*. Após um período no *Solutions* fui convidado a participar de uma nova empreitada em outro time.

Agora faço parte do Hotel team desde abril de 2022, onde passei a ocupar o cargo de *Associate Software Engineer 4*. Nesse time trabalhamos primariamente no projeto de *Customers Upgrade* da engenharia da empresa. Sirvo como base do time juntamente com meu Líder Técnico e minha gerente de Projeto aqui no Brasil e um colega desenvolvedor que se encontra na Califórnia. O time conta com uma rotação de novos integrantes para ajudar em demandas específicas de acordo com a necessidade de conhecimento em cada etapa em diferentes projetos que atuamos. Minha atuação se dá principalmente no *upgrade* de clientes, focado na área de *backend*, atuo também na tomada de decisões sobre os processos que iremos executar em cada cliente, juntamente com os outros integrantes da base do time, mantendo a documentação de nossos processos atualizada e também dando suporte ao resto do time a qualquer problema que estejam enfrentando.

Antes o processo de *upgrade* quando contratado por um cliente era feito exclusivamente pelo time de consultoria da empresa, conhecido como *Global Services*. Departamento que presta consultoria a diversos clientes que a *Liferay* possui globalmente, sendo um deles o serviço de *Upgrade*. Onde era assinado, por parte do cliente, uma contratação do *upgrade* da versão da sua instância do *Liferay portal* e as devidas adequações de contexto e correções. Um valor de contrato é baseado em cima das horas trabalhadas de um consultor, com um *upgrade* não é diferente, buscando competitividade comercial a *Liferay* vem se reinventando para minimizar os custos para os clientes e maximizar a produtividade dos desenvolvedores. As tecnologias envolvidas nessa evolução através do projeto *Upgrade* serão detalhadas a seguir.

3 Desenvolvimento realizado na empresa

Esta seção descreve como o *Hotel Team* encabeçou a implementação do processo de *upgrade* na engenharia da *Liferay* objetivando tirar essa demanda custosa do time de consultoria. Um processo que vem sendo repensado por toda a equipe, a qual me incluo desde seu início participando ativamente das várias etapas de decisão do nosso time de desenvolvedores. Para melhor compreensão das etapas, esta seção encontra-se dividida entre: a problemática e a solução proposta, as tecnologias utilizadas, e a contribuição.

3.1 A problemática e a solução proposta

A *Liferay* adquiriu diversos clientes ao longo de sua existência, passou por diversas formatações e reformulações, mas os clientes mais antigos contrataram o serviço baseados em um modelo *PaaS (Plataform as a Service)*. Cada cliente adquirido contratava o *Liferay DXP (Digital Experience Plataform)* para a sua versão mais atual do produto naquele momento. Em uma reunião semestral a empresa relatou a baixa migração desses antigos clientes para a versão mais recente do portal, possuindo ainda diversos clientes com versões antigas do produto. Isso acaba gerando para a empresa diversos custos e problemas.

3.1.1 Problemas e Justificativas

Um desses problemas é o alto custo de manter essas versões antigas de seu produto operacionais. As versões mais antigas apresentam sensíveis diferenças na arquitetura de software, nas tecnologias empregadas e em suas dependências de software. Algumas das versões mais antigas que ainda possuem clientes ativos são versões como a 5.x.x, enquanto a versão mais recente sendo é a 7.4.

Outro problema é o de *backport* de *bugs*, *bugs* encontrados e corrigidos em cada uma das versões podem ocorrer em versões mais antigas. Então existe um processo de teste do cenário do *bug* em questão em cada versão antiga que possua a funcionalidade afetada, e se o mesmo *bug* for encontrado, é necessária a análise caso a solução já encontrada é aplicável naquele contexto (com aquelas dependências, classes disponíveis, etc) e se não for, é necessário encontrar novamente uma solução para esse problema, agora nesse novo contexto.

O processo de *upgrade* é algo complexo e demorado, é necessário atualizar todas as dependências, referências, injeções que antes eram utilizadas no código, principalmente porque o *core* do *Liferay* Portal que antes era referenciado, seria modificado e não necessariamente permaneceria com os mesmos métodos e lógicas. Quando esse processo era principalmente executado pelo time de consultoria da *Liferay* e pago em horas trabalhadas, isso se tornava extremamente custoso para os clientes, por conta de diversos problemas que o time de consultorias enfrentava durante a execução desse *upgrade*. Como consequência disso, ocorria uma grande recusa por parte dos clientes em migrar seus produtos para uma versão permanecendo sempre na versão inicial que o produto foi montado, ou se feito algum *upgrade* anteriormente continuando naquela versão por anos sem atualizações.

Portanto, a *Liferay* em busca da aquisição de mais clientes e de melhor posicionamento no *ranking* global resolveu passar por uma mudança de visão estratégica. Embora ainda mantenha os clientes no esquema *PaaS* e ainda exista esta oferta, iniciou-se um processo de mudança do modelo de arquitetura de nuvem para *SaaS - Software-as-a-Service* com o lançamento do *LXC (Liferay Experience Cloud)*, uma reformulação da versão paga de seu produto com integração com seus outros produtos. É também clara a intenção de ter cada vez mais clientes para essa versão, seja através de novos clientes ou de migrações de antigos clientes do modelo *PaaS* para o modelo *SaaS*.

Os problemas para manutenção de versões antigas em estado operacional, o custo de manter e dar suporte a clientes nessas versões e a execução do *backport* de *bugs*, impuseram o desafio

de trazer o maior número de clientes possíveis para a versão mais moderna do código do produto, a 7.4, mesmo que ainda em *PaaS*, reduzindo aos poucos o custo sobre essas versões antigas com cada vez menos clientes. No melhor cenário trazer os clientes direto para o LXC no modelo *SaaS*.

O cliente estando no *SaaS* aproveitaria a possibilidade de passar por todos os *updates* do portal sem sentir o impacto, parar seu ambiente ou sequer ter que pagar para cada um deles pois uma das vantagens do *SaaS* é essa aplicação de atualizações na versão do portal que estiver na instância *SaaS*, sem afetar o produto. Mas como fazer isso se os clientes se recusavam a passar por esse complexo e caro processo de *upgrade* e como deixar esse processo menos custoso e demorado para ser executado pelo time de consultoria?

3.1.2 Solução

A *Liferay* trouxe uma solução que impactaria de alguma forma cada um desses problemas. Foi então criado no *Hotel Team*, o *Customers Upgrade*, projeto de *Upgrade* ao qual faço parte desde o começo, sendo agora o processo de *upgrade* também feito na engenharia e não somente nas consultorias. Qual a grande vantagem desse processo começar a ser executado na engenharia? O cliente não precisa mais pagar um contrato de horas trabalhadas na maior parte do processo, pois é um time de desenvolvedores da própria empresa executando o processo. Tínhamos muita coisa a aprender também, era a primeira vez que um time da engenharia realizaria o *upgrade* de um cliente e os processos envolvidos. Para atrair os clientes para embarcar nessa nova aposta, a *Liferay* fez a oferta desse *upgrade* a um custo bastante atrativo para os clientes que desejassem fazer o *upgrade* desta forma.

O *Hotel Team* e seu projeto *Customers upgrade* tinha que ter algo a mais para ajudar a empresa a reduzir esses custos de migração, afinal este trabalho deixaria de ser pago por clientes da consultoria e passaria a ser ativamente bancado pela própria empresa. Surge, então, o foco principal do *Hotel Team* com este projeto: melhorar a qualidade de vida dos desenvolvedores *Liferay* no executar de suas tarefas. Entrega-se para a consultoria o produto migrado para a versão mais recente, atingindo os pontos antes citados, mas durante esse processo é nossa missão elencar pontos de melhoria que podem ser executados nos processos relacionados ao ato de executar um *Upgrade* do produto de um cliente.

A melhoria na qualidade do desenvolvimento acontece quando é possível identificar pontos que se repetiam nos processos em cada projeto de *upgrade* que os diversos times de consultoria trabalhavam, pontos que eram resolvidos manualmente pelos desenvolvedores, e a partir disso trabalharmos para automatizar esses pontos. Desta forma, durante os primeiros projetos de migração foi possível ver o quão difícil esta tarefa se torna para os consultores que contam com prazos extremamente apertados, além não poderem contar com automação ou documentação que lhes auxiliem a ver erros similares. Por isso que foi pensado na catalogação de padrões que se repetem neste tipo de projeto, e na montagem de *scripts* para realizar certas tarefas.

Assim o processo de *upgrade* na engenharia poderá se tornar cada vez mais automatizado até só restar decisões e modificações que exijam um desenvolvedor para executar, reduzindo assim o custo com funcionários necessários para trabalhar nessas migrações e reduzindo o tempo para essas migrações serem entregues. Em um futuro com todos os padrões identificados e automatizados em

todas as versões legadas que o *Liferay Portal* possui, o processo de *Upgrade* otimizado será um padrão para a empresa, inclusive nos times de consultoria da empresa.

Outro motivo para esse projeto ser criado na engenharia é a ferramenta de otimização *sourceFormater*, de uso interno na *Liferay*. É nesta ferramenta onde os *scripts* de automação no processo de *upgrade* são desenvolvidos e adicionados, nós do *Hotel Team* contribuímos com o time responsável pela ferramenta na adição dessa nova *feature*. Então com a montagem e execução desse *backlog* de automação, esse fluxo de *upgrade* acaba se tornando cada vez mais otimizado e podemos atender mais clientes em um curto período de tempo.

3.1.3 Cenários de ofertas para clientes

A proposta ofertada, do *Customers Upgrade* na engenharia, para os clientes foi aperfeiçoada ao longo dos meses durante a execução do projeto, mas atualmente trabalhamos com as seguintes propostas:

No primeiro cenário, o mais comum, o cliente aceitou a oferta do *upgrade* e o mesmo seria executado pelo nosso time na engenharia. O nosso time na engenharia se encarregaria do *upgrade* de código e toda a migração do produto do cliente, independente da versão que estivesse, para a versão mais recente do portal (7.4), se certificando que todas as *features* do cliente estejam funcionando da mesma forma ou identificando substitutos *Out of the box* no próprio *Liferay Portal*. Após o *upgrade*, o próximo passo é colocar o produto do cliente, agora na versão 7.4, em produção. Como o foco do nosso time é na parte de *upgrade*, o time de consultoria responsável por esse cliente fica encarregado por essa etapa, o trabalho é então repassado para o time de consultoria faltando somente a finalização da entrega, reduzindo muito o esforço necessário a ser executado para o produto entrar em produção.

O segundo cenário é uma ramificação do primeiro, onde começamos o *upgrade* mas executamos somente até a primeira fase do processo. Nosso time executa até a fase de resolvermos todos os erros de compilação do cliente após o início de migração e entregamos então para o time de consultoria continuar a partir daí. Esse cenário geralmente acontece quando o cliente tem alguma espécie de restrição aos seus dados, dados sigilosos, concordou com o *upgrade* de seu produto nessa nova oferta, mas não com a disponibilização de uma versão de seu banco de dados para um time da engenharia. Sem acesso ao banco de dados nós não conseguimos continuar o fluxo de *upgrade* e passamos para o time de consultoria responsável pelo cliente, pois eles já estão sob o contrato de sigilo e têm acesso ao banco do cliente.

Esses cenários são ofertados tanto para clientes que optaram por permanecer ainda em *PaaS*, mas na versão do código mais recente, quanto para clientes que aceitaram a proposta de migrar para o *SaaS* com o *LXC*. A idealização do projeto foi feita para ser feito na engenharia e seria um grande desafio, seria a primeira vez que isso seria feito por um time da área de desenvolvimento da empresa e não de consultoria. Entro então nesse escopo com minha atuação no time, com o início do projeto de *Customer's Upgrade* planejado, após o aceite da oferta de um grande cliente brasileiro. Foi um *upgrade* no primeiro cenário e que iria direto para o LXC.

3.1.4 Fluxo de upgrade do projeto

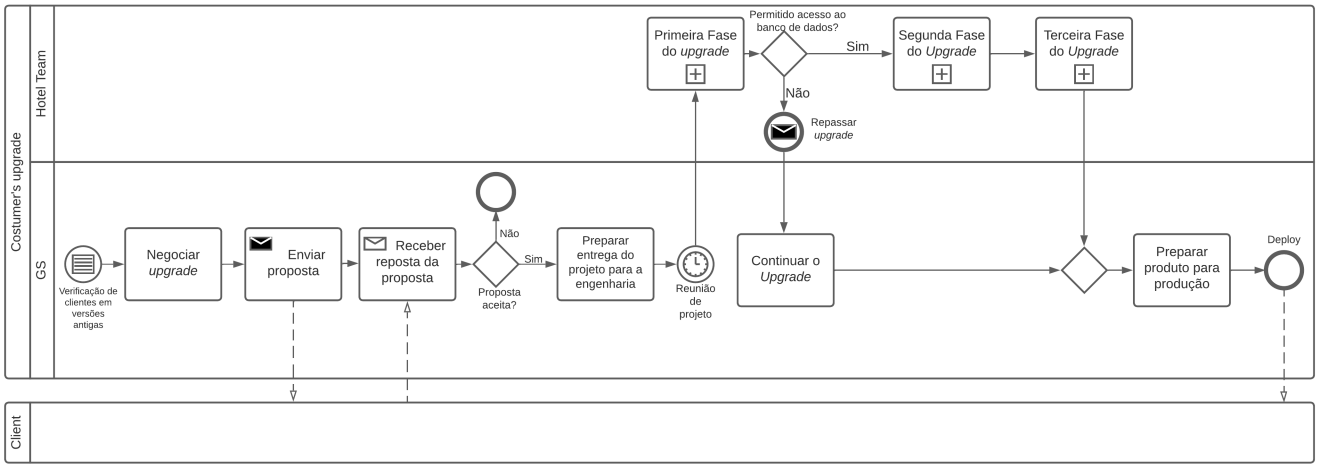

Figura 1: Fluxo geral do processo de Upgrade na engenharia [O AUTOR 2022a]

Para este primeiro cliente seguimos um fluxo de processos divididos basicamente em três fases principais, como podemos ver em 1, que haviam sido planejadas por Brian Chan, Idealizador do projeto, experimentamos esse fluxo, adaptamos e validamos nossos processos. Por ser nossa primeira tentativa, precisávamos ir experimentando muita coisa, analisando o que de melhor e pior aconteceu para aplicarmos nos projetos seguintes as melhores práticas.

A primeira fase, conhecida pela correção dos erros de compilação do código do cliente, é a fase onde atualizamos o *Liferay Workspace* do cliente. O *Liferay Workspace* é um ambiente gerado que é construído para dar suporte ao desenvolvimento de projetos *Liferay*, fornecendo vários *scrips Gradle* e propriedades pre-configuradas. Os projetos de nossos clientes são geralmente construídos utilizando esse ambiente e é a forma oficial de desenvolvermos módulos para um produto *Liferay*. Após atualizarmos temos as ferramentas necessárias para desenvolver um código para um *Liferay Portal* 7.4.

A *Liferay* atualmente lança atualizações com *releases* semanais, foi iniciado tal fluxo na versão 7.4.13 do *Liferay Portal* e o *update* fica anotado com o prefixo 7.4.13-uXX seguido do número do *update* semanal em questão 7.4.13-u37, por exemplo. Esses *releases* semanais são para onde estamos migrando o código do cliente, é a versão mais recente do *Liferay Portal*, tendo uma nova versão cada semana, escolhemos sempre a última versão disponível ao iniciar um novo projeto.

Como visto na figura 2, após o *Liferay Workspace*, atualizamos a dependência nos módulos OSGi do código do cliente. Atualmente modificando no *gradle.properties* para qual o *release* do produto o *Liferay workspace* está utilizando todos os módulos que declararem dependência em seus *build.gradle* irão consumir essa versão. Em versões anteriores do portal era necessário declarar em cada módulo a dependência individual para cada módulo do portal, com a versão mais recente isso não é mais necessário, pois com os *releases* semanais a *Liferay* trabalha com único *'.jar'* gerado com todos seus módulos compilados dentro dele. De forma que parte do nosso trabalho é também fazer essa limpeza de declaração de dependências desnecessárias no *build.gradle* de cada módulo, deixando o código mais limpo e mais legível em suas dependências, pois agora somente é necessário declarar a dependência para o *'.jar'* do *upgrade* e já ter acesso a todo o código *Liferay*.

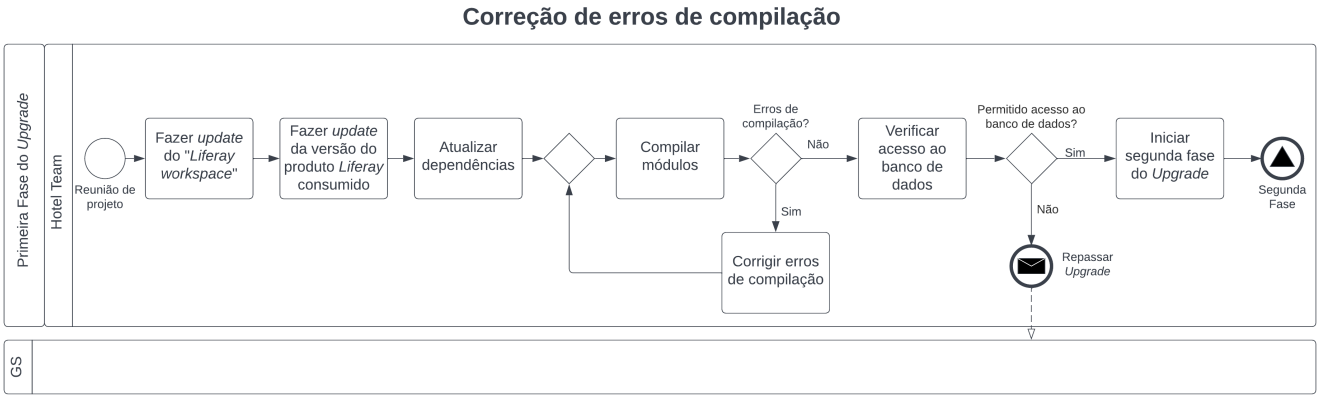

Figura 2: Primeira fase: Correção de erros de compilação [O AUTOR 2022b]

Chegamos então no fim dessa fase e no processo mais demorado, a correção dos erros de compilação. Como modificamos e atualizamos as referências do portal e de terceiros, boa parte do código que antes era utilizado pode ter sido alterado dependendo de quão longe a versão do cliente estava para a versão atual do Portal. Os erros mais brutos, como métodos que não existem mais, classes que mudaram sua assinatura, entre outros, são logo detectados ao compilarmos esses módulos. Com esse trabalho de atualização e compilação é feito um módulo por vez, nós então executamos o *build* de cada módulo e após isso temos no terminal o *log* de todos os erros de compilação que a classe em questão possui. Em seguida, cada um dos erros é corrigido e só após a conclusão destas correções a etapa é concluída.

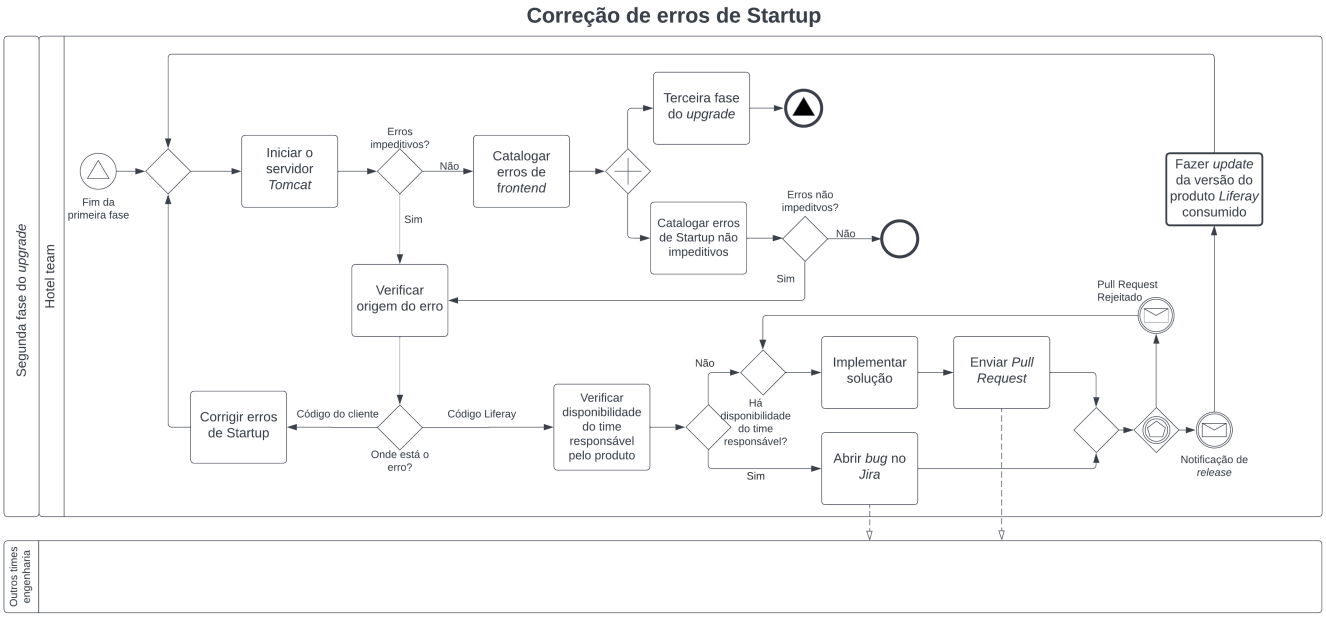

Figura 3: Segunda fase: Correção de erros de *Startup* [O AUTOR 2022c]

A segunda parte, demonstrada na figura 3, é conhecida como correção dos erros de *Startup*. Ela é a fase após termos todos os módulos compilando sem erros e todas as dependências resolvidas. Nós então executamos localmente o *script* responsável por levantar uma instância do *Liferay Portal*, o 'catalina.sh', chamado somente de Catalina. O Catalina é responsável por rodar as dezenas de *scripts* que um produto como o *Liferay Portal* precisa para subir seu servidor *tomcat* corretamente

[Sezov 2008]. O *script* começa realizando a inicialização do servidor catalina. Após todo o processamento ser executado ele exibe uma mensagem indicando que que o catalina foi inicializado com sucesso. Essa etapa é conhecida pela correção de erros de *Startup*, pois são erros que ocorrem durante um *upgrade* e impedem a inicialização do portal corretamente. Após a inicialização o servidor conecta no banco de dados do cliente que encontra-se hospedado localmente e começa a execução de diversos *scripts* de *upgrade* de banco.

Essa etapa atualiza índices, *schemas*, tabelas, *constraints*, e tudo ocorre durante a inicialização do servidor. Uma das etapas para realizadas é a checagem da versão do banco que está sendo consumido. Cada módulo do *Liferay Portal* também implementa uma classe responsável pelo mapeamento de mudanças na sua respectiva entidade, ao longo das diversas versões. Os *scripts* basicamente veem qual a versão atual no banco de dados, e rodam todos os *scripts* dessa versão em diante, de forma que ao fim dos *scripts* daquele módulo, aquela entidade estará na versão mais atual do *Liferay Portal*, dentro da base de dados do cliente. Esse processo é detectado e feito automaticamente ao tentarmos subir o servidor do catalina, mas pode ser executado separadamente por um *script* a parte, caso necessário.

Durante a inicialização também ocorre a compilação de todos os módulos do cliente e compilação dos temas que o cliente possa ter implementado em seu código. Um *Liferay Theme* é o aspecto geral de um site no portal *Liferay*. Os temas são uma combinação de CSS, JavaScript, HTML, e FreeMarker *templates* [LIFERAY 2022f]. Você pode fazer uso dos temas padrão, ou pode desejar criar o seu próprio *look and feel* personalizado para o seu *site* [Sezov 2011]. Após a compilação sem nenhum erro, vem a execução do *script* responsável por fazer *deploy* de cada um dos módulos OSGi do cliente para dentro do servidor *Tomcat* do catalina. Quando todos os *scripts* de inicialização rodarem sem erros e todas as etapas forem concluídas o servidor *Tomcat* estará funcional e pode ser acessado através do endereço e porta definidos.

Nessa fase temos que corrigir todos os erros impeditivos que encontrarmos durante a inicialização do servidor, muitos deles ocorrem no *deploy* dos módulos pois somente nessa etapa certos erros que não eram detectados acabam acontecendo, como *NullPointerException*, por exemplo. Outros podem ocorrer no *upgrade* do banco, as vezes o banco do cliente tem um cenário muito específico não mapeado pelos *scripts* de *upgrade* de alguma entidade específica. Faz parte do nosso trabalho depurar e identificar a causa de cada um dos erros de inicialização, corrigir os que forem de nossa responsabilidade e caso encontremos erros no código interno da *Liferay*, devemos endereçar para o time responsável por aquele produto, abrindo uma *task* no ambiente de acompanhamento de projeto *Jira*. Quando esse *bug* é corrigido, nós identificamos o *release* semanal que o mesmo foi lançado e atualizamos novamente o código do cliente para esta nova versão, de forma a testar se a correção resolveu o problema identificado no nosso cenário.

Existem diversos *bugs* que podem ocorrer nessa fase de inicialização, e cada um com um tempo de reparo distinto. Por conta disso, nós mapeamos todos os *bugs* encontrados, classificamos a prioridade entre eles e focamos em identificar a causa e definir uma possível solução para os impeditivos primeiro, de forma a quando tivermos todos os *bugs* impeditivos solucionados, a instância do *Liferay Portal DXP* do cliente vai estar rodando corretamente localmente e vai poder ser acessada pelo navegador. Com a instância do portal funcionando no navegador é possível paralelizar o trabalho e dar início a terceira fase.

Como podemos ver na figura 4, a terceira e última fase é onde ficamos responsáveis por solucionar os erros do *frontend* e ocorre em paralelo ao fim da segunda fase para finalizar os erros de *backend* que foram mapeados anteriormente e ainda não foram solucionados. Quaisquer novos erros de *backend* que forem aparecendo no decorrer das correções de *frontend* também são tratados. Nessa fase a instância do *Liferay Portal* do cliente já está rodando localmente e podemos acessar as páginas e verificar os fluxos de navegação do portal, se tudo está funcionando como deveria. Nós deixamos uma instância da versão original do Portal do cliente para podermos fazer as comparações de como as telas, funções, etc, deveriam estar depois do *upgrade*. O objetivo é deixar a atualização mais próxima possível do visual e funcionalidades anteriores. É possível que o cliente tenha na versão original uma *feature* do *Liferay Portal* que já foi depreciada com o tempo e substituída por outras. Desta forma, é necessário fazer o mapeamento dessas mudanças obrigatórias que acabarão acontecendo e repassamos para o time de consultoria, para validarmos se a nova *feature* atende ou não a necessidade do cliente.

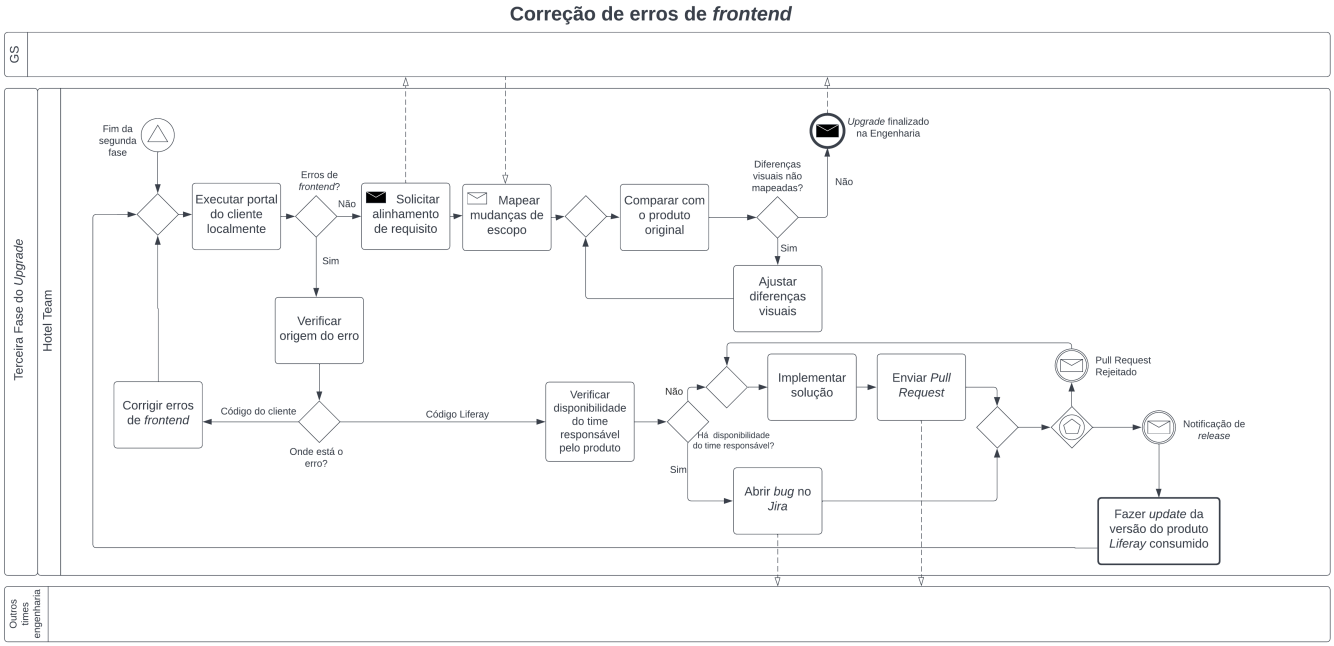

Figura 4: Terceira fase: Correção de erros de *Frontend* [O AUTOR 2022d]

Enquanto os integrantes do nosso time responsáveis pelo *frontend* analisam os problemas relacionados a tela, posicionamento, comportamento, etc. O time de *backend*, ao qual faço parte, foca em eliminar o restante dos erros que ainda estiverem ocorrendo na inicialização. Em seguida, o time concentra-se na correção dos erros que acontecem ao navegar pelo site, chamadas a métodos específicos que não estão funcionando e arquivos *Freemarker* que referenciam códigos que foram removidos, por exemplo. Vale salientar que nosso foco não é melhorar o produto do cliente, e sim realizar o *upgrade*. O erros existentes antes do processo de *upgrade* são conservados visto que estão fora do escopo do time.

Os principais erros desta fase estão concentrados nas mudanças das estruturas padrões usadas no portal *Liferay* que o cliente antes utilizava de uma forma e que a usabilidade mudou ao longo do tempo. Erros nos arquivos *'css'*, *'js'*, *'Freemarker'* são bastante comuns também, pois só são possíveis de serem detectados nessa última fase. Participei ativamente da indentificação e indentificação de *bugs* nos produtos da própria *Liferay*, ao realizar essa identificação aplicamos as *releases* que

contenham as correções e esses novos *updates* podem ocasionar novos problemas em qualquer uma das fases. As *releases* semanais, em geral, geram menos falhas pois tratam-se de mudanças menos complexas do que as antigas mudanças de versões do *Liferay Portal*. Após todos os problemas identificados serem corrigidos e mudanças autorizadas, nós fazemos a entrega para o time de consultoria responsável pelo cliente continuar o fluxo até o código, agora já migrado, entrar em produção.

3.1.5 Atuação no time

Atualmente já trabalhamos e entregamos para o time de consultoria, *Global Services (GS)*, 3 projetos de *upgrade* e já estou atuando no quarto, com mais alguns já previstos para entrarem na nossa fila de trabalho. Participei ativamente de todos esses trabalhos, inclusive da definição de processos e fluxos a serem seguidos em cada fase. A definição de um processo organizado para o trabalho foi importante, pois por ser uma empreitada completamente nova para a engenharia não havia um modelo a ser seguido. Inicialmente, tínhamos a definição de que seriam 3 fases principais, porém não existia nada definido para como executar cada uma dessas fases de forma que o produto estivesse pronto dentro do prazo estabelecido inicialmente.

O nosso time é composto fundamentalmente por 4 integrantes fixos, sendo dois desenvolvedores de *backend*, um líder técnico que também trabalha com o *backend* e uma gerente de projetos. Os demais membros são membros rotativos, com a rotação funcionando da seguinte forma: em cada fase após começarmos a executar nossas tarefas, podemos mensurar a quantidade de tarefas que precisarão ser executadas, se forem muitas e precisarmos de ajuda ou forem fora da nossa *expertise*, são alocados membros temporários para nos ajudar na entrega de cada *upgrade*. Acontece principalmente na fase de *fix* de *frontend issues*, pois só temos desenvolvedores de *backend* na base do projeto. Desta forma, sempre na terceira fase, temos desenvolvedores de *frontend* atuando paralelamente às correções do ambiente de *backend*.

Durante as fases eu consegui diagnosticar problemas que eram difíceis de serem encontrados em fluxos de *upgrade* que já existiam há anos, participei da estruturação do ambiente de desenvolvimento de novos membros que entravam para ajudar no projeto, ajudando os mesmos a se situarem e estarem aptos a trabalhar, sempre com a ajuda do meu líder técnico me dando o direcionamento necessário. Durante o primeiro Projeto de *upgrade*, eu participei do trabalho de identificação de padrões nas correções que estávamos executando em que as soluções pudessem ser automatizadas, correções que não necessitavam de tomada de decisão, que eram um trabalho puramente manual e repetitivo entre os módulos, e que provavelmente se repetiriam em qualquer outro projeto de migração a partir dessa mesma versão, ou até versões posteriores. Com esse mapeamento sendo feito e identificado em uma documentação do nosso time, de forma que no próximo projeto, uma parte considerável dos problemas encontrados já tinha sido mapeada e podíamos somente replicar a solução mapeada para cada um dos problemas já identificados agora no novo projeto, e com o novo projeto adquirimos ainda mais padrões de *bugs* e suas soluções propostas.

A documentação elaborada durante os processos de *upgrade* estavam sendo bastante úteis, no entanto, não encontravam-se adequadamente organizadas de modo a otimizar nossa produtividade. Por esta razão eu desenvolvi um trabalho onde identifiquei padrões dentre os problemas encontrados,

agrupei-los, removi informações redundantes em diversos pontos do documento e fiz a classificação entre problemas de *backend* e de *frontend*. Dentro destas categorias eu criei subdivisões para cada um dos problemas que tinham como ser automatizados e problemas cuja solução não era automatizável, mas que deveriam ser mapeadas pois poderiam se repetir em projetos seguintes. Definindo também padrões de escrita, marcação, agrupamento e formatação do documento. A estrutura proposta tornou-se padrão para a documentação dos mapeamentos. Desta forma, cada versão antiga de portal que pegamos para fazer um *upgrade* recebe um documento criado a partir deste modelo proposto. A partir desse documento foi montado o *backlog* de histórias de automação, histórias a serem desenvolvidas no *sourceFormater* para melhorarmos esse fluxo de *Upgrade* de código de clientes *Liferay*.

3.2 Tecnologias utilizadas

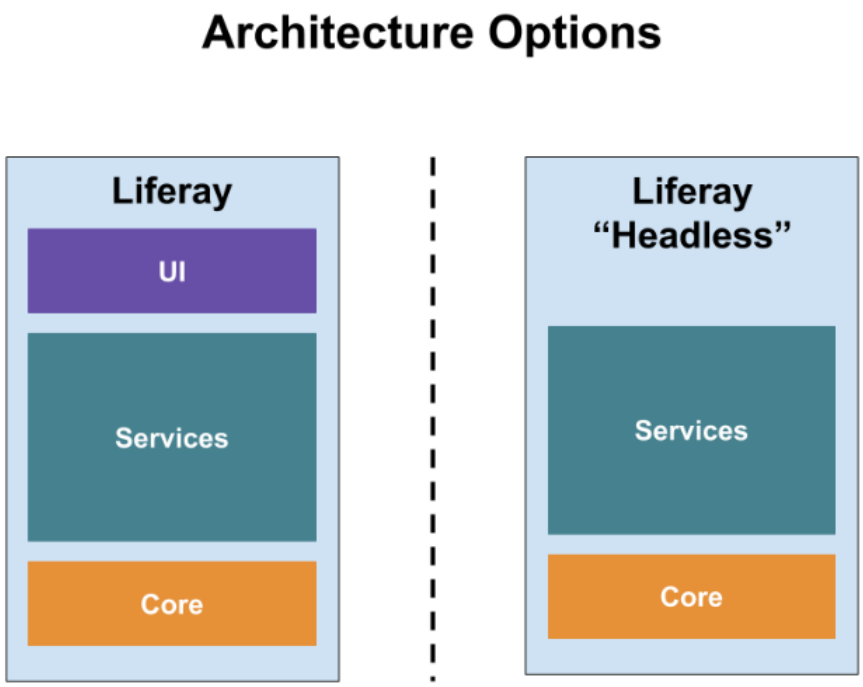

Figura 5: Opções de arquitetura *Liferay Portal* [LIFERAY 2022b]

Neste tópico, será discorrido sobre as tecnologias que integralizam o projeto de *upgrade* do time *Hotel*. A arquitetura do *Liferay Portal* é dividida em três partes principais, como pode ser visto na figura 5.

O *Core* provê a *runtime environment* para gerenciamento dos serviços, *frameworks*, *UI components*, e customizações no Portal. Alguns de seus componentes são serviços, serviços customizados, *Language keys* (mensagens localizadas pelo idioma), *portlets*, entre outros [LIFERAY 2022b].

A camada de serviços, expõe as *features DXP* e a *features* personalizadas através de *APIs Java* e *APIs web* [LIFERAY 2022b]. A lógica empresarial é implementada em serviços instalados no *runtime*

environment dos componentes. Os serviços *core* incorporados e os serviços de *frameworks* operam em *models* como *Users*, *Roles*, Conteúdo Web, e muito mais. É possível introduzir novos modelos e funcionalidades através de serviços personalizados. Os componentes dos serviços podem acessar uns aos outros através de injeção de dependência.

A *UI (User Interface)* não é obrigatória, pode-se utilizar a UI e os *Services* em conjunto ou se concentrar somente na utilização de *Services* através de *APIs REST* ou de *APIs GraphQL*. Mas a mesma se utilizada ajuda as pessoas a trabalhar, colaborar, e utilizar os conteúdos adicionados no produto. Ela consiste em uma aplicação DXP web para adicionar portais, *sites*, páginas, *widgets*, e conteúdo no produto. Aplicações onde *widgets* fornecem uma interface de usuário para os *services* já instalados. E por fim os *Liferay Themes*, temas são *plugins* para estilizar sites com um *look and feel* único.

O *Liferay Portal* possui diversas tecnologias em todo o seu sistema, não atuamos em todas as camadas no contexto de *upgrade* mas uma das principais que atuamos são as suas modularizações em cima do *OSGI framework*. *OSGi* define um sistema modular e dinâmico para Java, define uma arquitetura para o desenvolvimento e implantação de aplicações modulares e de bibliotecas [Ahn, Oh e Sung 2006]. O *Liferay Portal* tem suas *features* modularizadas permitindo que cada um deles declare suas dependências e exporte seus próprios *java packages* caso queria que sejam consumidos por outros módulos da aplicação [LIFERAY 2022c].

Durante o processo de *upgrade* é fundamental entender a estrutura de referência dos módulos para que seja possível rastrear onde um *bug* possa estar acontecendo no fluxo. Há fluxos que apresentam grande complexidade exigindo maior tempo de estudo dos módulos para que a solução para aquele caso único possa ser encontrada. A estrutura de um módulo varia de acordo com seu propósito, se vai ou não possuir interface, etc. Um exemplo de arquitetura seria possuir um pacote *web*, pacote de *API*, pacote *service*, pacote de implementação desses serviços, e um pacote de testes para aquele módulo.

Os módulos precisam ter implementados em sua camada interna um pacote de *upgrade*, é nele que as classes java responsáveis pelo *upgrade* irão verificar a versão atual do banco do cliente para aquele módulo comparando com a última versão disponível, caso haja a diferença, irá executar os passos definidos na implementação daquele módulo da classe *UpgradeStepRegistrator*. Cada versão daquela entidade no banco de dados possui um *UpgradeProcess* implementado, como pode ser visto na Figura 6 , a classe java responsável irá executar o método *doUpgrade* para cada versão que esteja faltando no banco do cliente até aquela entidade estar igual a sua última versão do portal. Esses passos de *upgrade* são responsáveis tanto por modificar a estrutura da tabela quanto por popular as novas colunas com as informações necessárias e certas para cada registro na tabela. São nesses passos que geralmente podem ocorrer alguma inconsistência nos dados ocasionando um erro na fase de inicialização do *Liferay Portal*.

Uma camada em que atuamos ativamente na terceira fase do *upgrade* é na *UI* do sistema do cliente, especificamente na estilização de suas páginas, no tema de seu site. Um tema torna possível criar um *look and feel* único para cada cliente. É desenvolvido como um projeto *WAR (web archive)*, que inclui *CSS*, *JavaScript* e conteúdos de marcação como *Freemarker*, por exemplo. Os erros principais acabam acontecendo dentro dos arquivos *Freemarker* do cliente, pois os mesmos muitas vezes fazem referências ao código *core* do *Liferay Portal* que foi modificado entre versões. Erros de

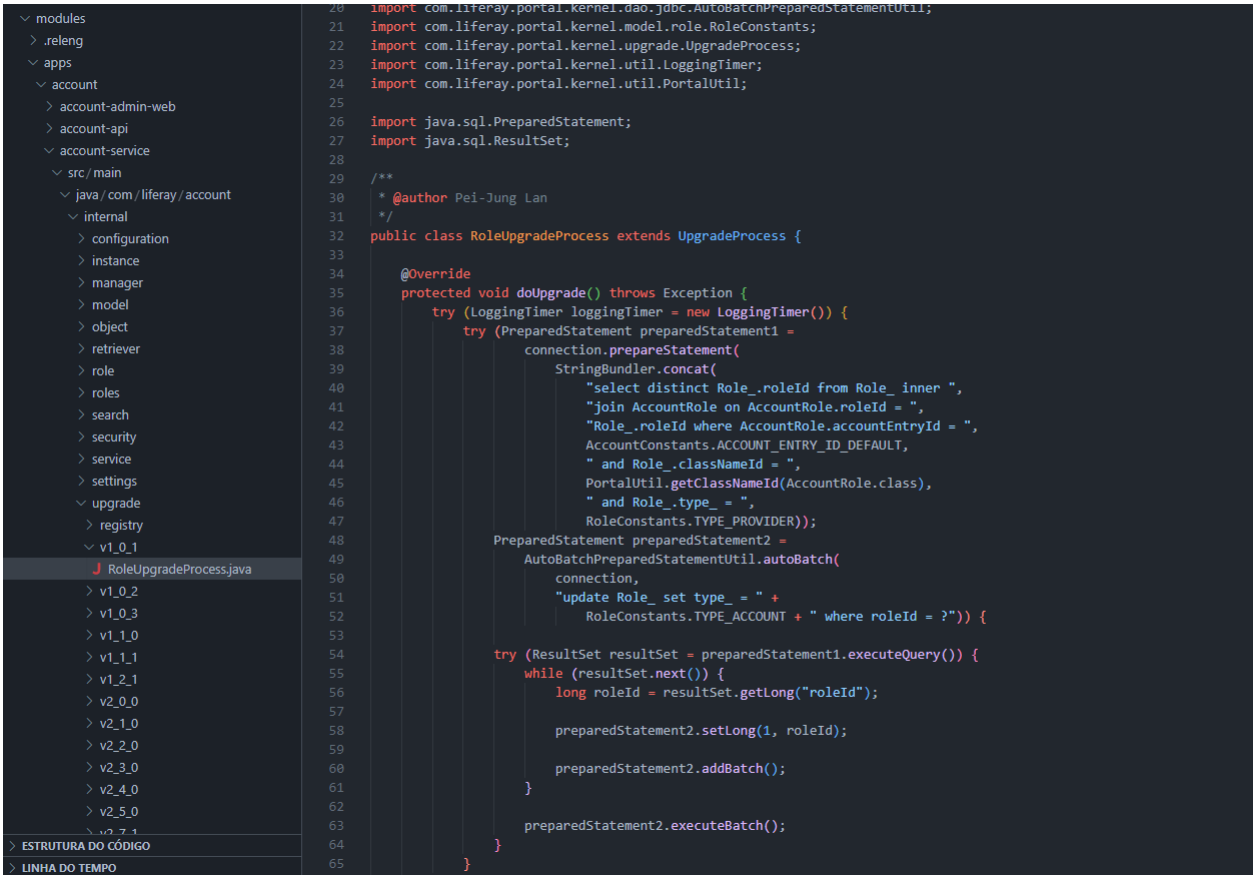

Figura 6: *RoleUpgradeProcess.java* no código fonte do *Liferay Portal*)

'CSS' e *'js'* também precisam ser tratados por conta das mudanças de sintaxe entre versões e da implementação das bibliotecas.

3.3 Contribuição

No time *Hotel* participei da detecção de *bugs* claramente específicos de produtos *Liferay*, pois os cenários para identificar-los são extremamente específicos à tecnologia aqui desenvolvida. Cenários estes de difícil detecção durante o fluxo de *upgrade* e complicada reprodução em testes, por isso foi gerada a necessidade de repassar os *bugs* para o time originalmente responsável por cada produto. No entanto, por vezes os times já estavam tomados de demandas em sua própria fila de desenvolvimento, não sendo possível adicionar à ela nosso pedido de correção de *bug* em um futuro próximo. Então por diversas vezes eu fui responsável por implementar tais correções, com isso nós mesmos durante um processo de *upgrade* melhoramos o produto *Liferay*, a funcionalidade e *upgrade* dos produtos internos.

Desde que começamos em abril de 2022, já participei da finalização de quatro clientes de *upgrade* da engenharia, os outros três já foram entregues aos respectivos times de consultoria responsáveis e estão a caminho de entrarem em produção. Eu fui capaz de ajudar o time com a definição de um padrão de documentação e coleta de melhorias e problemas conhecidos no processo de upgrade, com essa coleta as nossas próprias entregas começaram a ficar cada vez mais rápidas, principalmente pelo tempo de conclusão das primeiras fases onde foram encontrados mais possíveis automações.

Em cada projeto que pegamos conseguimos extrair mais dados para esse *backlog* de automação, o que vai deixando o nosso processo de *upgrade* cada vez mais otimizado.

E essa é uma das principais contribuições do meu trabalho e do meu time para a empresa. Nós conseguimos melhorar o processo de atuação dos desenvolvedores do time de consultoria no cenário de *upgrades*. Quando nosso trabalho for finalizado, o fluxo de processos e *scripts* passarão a ser o novo padrão para os processos de *upgrade*, reduzindo assim o tempo necessário para projetos desse tipo serem entregues, a quantidade de horas necessárias de desenvolvimento, etc.

Durante nosso período desenvolvendo o novo formato do *upgrade* do *Liferay Portal* trabalhei trazendo diversos clientes antigos para novas versões do *Liferay* no decorrer deste projeto. Por si só, esta atividade tem grande impacto na manutenção desses clientes. Na nova versão, esses clientes estão voltados para uma maior facilidade de aplicação de *updates* que agora são semanais, assim atendemos também ao desejo da *Liferay* de trazer o máximo de clientes possível para o 7.4. Juntamente com a melhoria do processo em si de *upgrade* quando todos os novos projetos de *upgrade* da consultoria passarem a seguir a nova forma, toda oferta feita para clientes vai se tornar mais atrativa, seja em tempo de entrega ou até mesmo o custo a ser pago pelo cliente.

4 Dificuldades encontradas

Acredito que a primeira dificuldade que tive foi a de adaptação, além de ter acabado de entrar em um novo projeto, havia saído de uma área de desenvolvimento de *features* propriamente dita para tratar de *upgrade* de clientes do *Liferay Portal*. Eu nunca havia lidado com uma versão além da *Master* do Portal, não sabia quais eram as diferenças entre versões e como se dava o processo de *upgrade* de cada parte do *Liferay Portal*. A minha nova gerente e meu novo líder técnico me auxiliaram bastante na minha integração no time e nessa adaptação, podendo o quanto antes começar a produzir mais individualmente.

Precisei ir evoluindo a minha capacidade de identificar o local exato onde o problema estaria no enorme fluxo de chamadas de métodos que o Portal possui e a medida que fui consertando cada vez mais problemas nas diversas fases de *upgrade* fui adquirindo conhecimento para quando o mesmo problema ou problemas parecidos ocorressem ao longo dos projetos eu já saberia identificar um caminho possível para a solução.

Além da dificuldade com a estrutura interna do código dos diversos produtos *Liferay*, tinha a falta de conhecimento a cerca dos produtos *Liferay* como um todo, na própria interface do portal, como eles se relacionavam, como eram utilizados, como era feita a montagem das telas do portal. Antes, por trabalhar em um projeto especifico, eu só tinha conhecimento a cerca de produtos que o produto que eu estava desenvolvendo teria integração, não era necessário um conhecimento tão completo de *Liferay*. Os módulos são planejados para serem desenvolvidos independentemente de forma a facilitar o desenvolvimento [LIFERAY 2022c].

Tive necessidade de melhorar meu conhecimento sobre como os módulos *Liferay* se relacionavam, entender mais sobre *OSGI*, *gradle*, diversas tecnologias que o código *Liferay* tem dentro de si. Durante as fases do upgrade muitas vezes precisamos recorrer a outros times da *Liferay* Global, tirar dúvidas, informar *bugs*, pedir ajuda e em grande parte isso era feito em inglês. Eu já possuía

um entendimento razoável de inglês, mas tinha certa dificuldade de me comunicar com outros times, principalmente de fora do país. Mas a necessidade me fez melhorar essa *soft skill* e minha conversação melhorou bastante com o contato as vezes diário com colegas de outras locações da *Liferay*.

Isso foi muito importante pois em períodos que meu líder técnico esteja ausente os demais membros do meu time contam exclusivamente comigo como o responsável pelo *backend* do projeto. Então foi necessário esse crescimento para que eu possa fazer esse trabalho de forma mais independente.

5 Impactos da sua formação no seu trabalho

A minha formação no curso de Bacharel em Sistemas de informação na Universidade Federal Rural de Pernambuco teve bastante importância nas atividades em que atuo/atuei durante meu tempo na *Liferay*. O curso possui uma metodologia de ensino baseado em problemas, ou *Problem-based Learning (PBL)*. Onde durante todo o curso temos diversos problemas para serem resolvidos, em formas de projetos, seja individualmente ou em grupos. Tendo a construção do conhecimento baseado na resolução de problemas [Wells, Warelow e Jackson 2009] o que fundamentou muito da minha capacidade de desenvolvimento e construção do meu raciocínio lógico.

Além da cadeira de Introdução a programação, ministrada pelo professor Dr.Cícero Garrozi, disciplina responsável por me ensinar o básico da lógico de programação e disseminar a vontade de aprender mais a respeito. A disciplina de laboratório de Programação ministrada pelo mesmo foi de extrema importância para o desenvolvimento do meu raciocínio lógico voltado para soluções de problemas, gostaria de ressaltar a forma que a cadeira foi ministrada pelo professor na época como responsável por isso.

Enxergo uma influência bastante relevante no que aprendi em Modelagem e Programação orientada a objetos, ministrada pelo professor Dr. Gabriel Alves. Nessa disciplina aprendi a fundamentação teórica e prática no desenvolvimento orientado a objetos, fundamental para meu entendimento de fluxos e estruturas de código já existentes ou que eu mesmo viria a desenvolver no portal *Liferay*. Nessa cadeira também tive meu primeiro contato com a Linguagem de programação Java, linguagem na qual o código do Portal foi construído. Além disso, foi com ela que tive um primeiro contato com uma estrutura de entrega e análise de código semanalmente. Essa experiência com o desenvolvimento de um produto, focando em qualidade, entregas semanais e utilidade para o cliente foi ainda mais explorada na cadeira de Análise e Projetos de Sistemas de Informação ministrada pelo professor Dr. Cleviton Vinicius.

A graduação serviu como base para o aprendizado a me desenvolver seguindo uma metodologia ágil, como desenvolver e analisar problemas em um banco de dados. Juntamente com diversas outras fundamentações que serviram de base para o trabalho que já desenvolvi em *Liferay* e meu trabalho atual no time de *upgrade*.

Mas eu acredito que ainda mais importante que a base que o Curso de Bacharelado em Sistemas de informação - UFRPE, tenha sido a minha experiência com desenvolvimento de produtos reais na realização de estágios não obrigatórios previstos como atividade complementar do curso.

As minhas experiências profissionais anteriores foram fundamentais para testar e desenvolver minha capacidade de identificar e resolver problemas em produtos reais, utilizados por clientes reais, entendendo melhor as dores e como implementar código focando cada vez mais em qualidade e na fácil manutenção.

O meu estágio na própria *Liferay* inclusive, foi uma experiência realmente enriquecedora nesse quesito, entendendo cada vez melhor diversos padrões de qualidade que só havia tido contato durante a faculdade. Padrões que já tinha até visto aplicados em outras experiências profissionais em estágios anteriores, mas que apresentavam falhas de planejamento ou falta de definição de padrão dentro do código. Foi de fato uma experiência enriquecedora fazer parte da concepção desse fluxo de trabalho e compreender como repensar a organização do desenvolvimento pode nos beneficiar como desenvolvedores.

6 Conclusão

Como desenvolvedor *Liferay* participei do desenvolvimento e manutenção de *features* no *Liferay Portal*, produto interno do Portal, o que por si só já agregou valor à minha atuação na empresa até então. Mudando um pouco de cenário fui convidado a fazer parte do Hotel, time da engenharia da empresa. Com um projeto bem inovador dentro da empresa: trazer também para engenharia a migração de código de clientes *Liferay*, projeto de *Customer's Upgrade*.

O time Hotel tem como principal ideal o de melhorar a qualidade de vida dos desenvolvedores *Liferay*, ajudá-los a executar seus trabalhos de forma mais fácil focando realmente na qualidade daquilo que está sendo desenvolvido e perdendo cada vez menos tempo com fluxos e processos desse desenvolvimento. E com esse projeto nós estamos reformulando e definindo um novo padrão para o *upgrade* de código de clientes dentro da empresa, tornando esse processo menos doloroso para quem for executá-lo com boa parte desse processo automatizado, reduzindo a quantidade de desenvolvedores necessários para a realização de um *upgrade* e até mesmo o tempo para que um *upgrade* passe a ser entregue por completo para o cliente.

Esse nosso trabalho tem impacto direto na atração e convencimento de clientes antigos *Liferay* a aceitarem essa oferta de *upgrade* reformulada trazendo-os para a versão mais recente do portal. Cada cliente que nós mesmos trouxemos para a versão mais recente do portal, ou clientes que serão trazidos no futuro graças a esse novo *upgrade*, impacta diretamente o custo da empresa em manter esses clientes operacionais em versões antigas do portal *Liferay*, o custo de *backport* de *bugs* e de horas trabalhadas por nosso time de consultoria nessas versões antigas do portal. Quanto menos clientes nessas versões antigas mais esses custos vão sendo aliviados possibilitando a empresa de focar cada vez mais em melhorias e novas *features* para o produto a serem lançadas semanalmente na versão 7.4 do *Liferay Portal*.

Este é um trabalho em andamento, o nosso time consegue absorver cada vez mais coisas a cada nova versão do Portal que precisamos realizar o *upgrade*. Irei começar a atuar também no desenvolvimento dos *scripts* de automação a serem executados nos *upgrades*. Estes *scripts* serão desenvolvidos na ferramenta *sourceFormat* da própria empresa. Alguns dos *scripts* já foram implementados e no próximo *upgrade* já devemos fazer uso dos mesmos. No futuro nosso time tem planos também de melhorar a ferramenta de *CI (Continuous integration)* da *Liferay*, focando em reduzir o

tempo necessário para a análise do código enviado para os repositórios *Liferay*.

Referências Bibliográficas

[Ahn, Oh e Sung 2006]AHN, H.; OH, H.; SUNG, C. O. Towards reliable osgi framework and applications. In: *Proceedings of the 2006 ACM symposium on Applied computing*. [S.l.: s.n.], 2006. p. 1456–1461.

[CHAN 2014]CHAN, B. *DevCon 2014 - Interview with Brian Chan, Co-Founder of Liferay*. 2014. Acesso em 03 de Outubro 2022. Disponível em: <https://www.youtube.com/watch?v=GmC0LNA-zII>.

[KON 1996]KON, A. *Evolução do setor terciário brasileiro*. São Paulo, 1996. Disponível em: <https://bibliotecadigital.fgv.br/dspace/handle/10438/3034?show=full>.

[LIFERAY 2017]LIFERAY. *Vida en Liferay*. 2017. Acesso em 03 de Outubro 2022. Disponível em: <https://www.youtube.com/watch?v=29i0pUYA1F8>.

[LIFERAY 2020]LIFERAY. *Liferay Latin America | 10 Anos de Aniversário (Long Version)*. 2020. Acesso em 03 de Outubro 2022. Disponível em: <https://www.youtube.com/watch?v=RM-A1yDqG0E>.

[LIFERAY 2021]LIFERAY. *2021 Gartner Magic Quadrant Names Liferay as Leader for Digital Experience Platforms*. 2021. Acesso em 03 de Outubro 2022. Disponível em: <https://www.liferay.com/pt/blog/liferay-experience/2021-gartner-magic-quadrant-names-liferay-as-leader-for-digital-experience-platforms>.

[LIFERAY 2022a]LIFERAY. *Casos de Sucesso*. 2022a. Acesso em 03 de Outubro 2022. Disponível em: <https://www.liferay.com/pt/resources/case-studies>.

[LIFERAY 2022b]LIFERAY. *Introduction to Architecture*. 2022b. Acesso em 03 de Outubro 2022. Disponível em: <https://help.liferay.com/hc/pt/articles/360035124151-Introduction-to-Architecture>.

[LIFERAY 2022c]LIFERAY. *OSGi and Modularity*. 2022c. Acesso em 03 de Outubro 2022. Disponível em: <https://help.liferay.com/hc/pt/articles/360035467532-OSGi-and-Modularity>.

[LIFERAY 2022d]LIFERAY. *Our Story*. 2022d. Acesso em 03 de Outubro 2022. Disponível em: <https://www.liferay.com/pt/company/our-story>.

[LIFERAY 2022e]LIFERAY. *Soluções personalizadas construídas com agilidade e segurança, conectadas em uma só plataforma*. 2022e. Acesso em 03 de Outubro 2022. Disponível em: <https://www.liferay.com/pt/one-platform-endless-solutions>.

[LIFERAY 2022f]LIFERAY. *Theme Components*. 2022f. Acesso em 03 de Outubro 2022. Disponível em: <https://help.liferay.com/hc/pt/articles/360035219072-Theme-Components>.

[Sezov 2008]SEZOV, R. *Liferay Administrator's Guide*. [S.l.]: Lulu. com, 2008.

[Sezov 2011]SEZOV, R. *Liferay in action: the official guide to Liferay portal development*. [S.l.]: Simon and Schuster, 2011.

[Wells, Warelow e Jackson 2009]WELLS, S.; WARELOW, P.; JACKSON, K. Problem based learning (pbl): A conundrum. *Contemporary Nurse*, Taylor & Francis, v. 33, n. 2, p. 191–201, 2009.