Itens de um Plano de Projeto

Histórico da Revisão

Data	Versão	Descrição	Autor(es)
11/04/2025	0.1	Planejamento Inicial	Cauê, Jesus, Letícia, Luiz, Pedro
24/04/2025	0.2	Primeiras Semanas	Cauê, Jesus, Letícia, Luiz, Pedro

1. Identificação

Nome do Projeto: OrSync
Equipe: AI-dentro
Data de criação do documento: 11/04/25

Cauê Paiva Lira - 14675416
Jesus Sena Fernandes - 12697470
Letícia Raddatz Jönck - 14589066
Luiz Felipe Diniz Costa - 13782032
Pedro Louro - 13672446

2. Introdução

Objetivos: Melhorar a integração entre a API e a plataforma do ORCID com outros portais amplamente utilizados no meio acadêmico para indexação de artigos científicos e armazenamento de dados e códigos utilizados na pesquisa.

Problema atual: Apesar de ser amplamente utilizado como identificador de pesquisadores, o ORCID apresenta uma experiência de uso bastante limitada. A interface é pouco amigável e torna difícil localizar informações relevantes sobre os pesquisadores, como formas de contato e publicações específicas por área de atuação. Não há filtros eficientes para navegar entre os conteúdos, exigindo buscas manuais e pouco práticas. Além disso, a ausência de recursos para gerar versões organizadas de currículos e a falta de indicadores claros sobre o impacto científico de cada pesquisador acabam tornando o sistema pouco funcional no dia a dia acadêmico.

Público-alvo: Professores e pesquisadores do ensino superior.

Propósito: Fornecer uma melhor experiência ao público-alvo, promovendo transparência, compartilhamento de informações e valorizando o impacto científico dos pesquisadores.

Evolução do plano: O plano irá evoluir baseado nas integrações escolhidas para o desenvolvimento final e também com o feedback do usuário coletado, baseado nesse **contexto fluído de desenvolvimento**, vamos adotar o modelo ágil de desenvolvimento

Modelo de desenvolvimento: Modelo **ágil**.

3. Escopo do projeto

Análise e refinamento do escopo

A primeira etapa do projeto envolve a análise das possibilidades de integração do ORCID com outras plataformas relevantes para a pesquisa e a vida acadêmica, como o **Lattes**, **Google Scholar**, **GitHub** e **GitLab**. Com base nessa análise, será realizada a seleção de uma a três integrações prioritárias, levando em consideração a aderência aos requisitos levantados junto aos stakeholders, a relevância e utilidade acadêmica, e a viabilidade técnica para implementação no período previsto do projeto.

Após a definição das integrações, iniciaremos um aprofundamento técnico para entender as particularidades de cada caso e refinar os requisitos específicos. Essa investigação orientará a divisão das tarefas entre a equipe, possibilitando que cada integrante se concentre em um escopo claro e alinhado ao cronograma de entregas.

Considerando o caráter exploratório do projeto e o contexto dinâmico do meio acadêmico, o escopo do OrSync permanecerá flexível ao longo do desenvolvimento. À medida que novas necessidades forem identificadas em testes com usuários e professores, ajustes serão incorporados para garantir que a ferramenta evolua de forma prática, relevante e conectada às reais demandas dos pesquisadores.

Requisitos Funcionais

1) Integração fácil e intuitiva com os portais externos estabelecidos, permitindo compartilhamento de informações.

2) Exibir claramente para o usuário quais dados foram extraídos de cada integração, promovendo total transparência no processo.
3) Implementar sistema de busca e filtragem de publicações por palavras-chave, áreas de atuação e tipo de material publicado.
4) Disponibilizar funcionalidade de geração automática de currículos personalizados, permitindo ao pesquisador escolher entre formatos resumidos (1 página) ou completos (incluindo lista de publicações e indicadores).
5) Construir dashboards de visualização de métricas, como citações recebidas, áreas de especialização e rede de coautoria.
6) Exibir formas de contato do pesquisador de forma organizada, com opção de integrar e destacar links para email, LinkedIn e websites pessoais.

Requisitos Não Funcionais

1) Segurança e Privacidade dos dados: As integrações devem respeitar leis de privacidade de dados (**LGPD**), com os dados extraídos sendo públicos.
2) Confiabilidade e compatibilidade: Integrações escolhidas devem ser estáveis, para garantir a compatibilidade com versões futuras do software, de modo a assegurar a confiabilidade da solução para o usuário final.
3) Documentação técnica: Manter documentação clara e acessível sobre a arquitetura do sistema, integrações realizadas, fluxos de dados e APIs utilizadas, visando facilitar a continuidade e a escalabilidade do projeto.
4) Resiliência a Falhas Externas: Implementar mecanismos de tolerância a falhas como *retries* automáticos com *backoff* exponencial, fallback seguro em caso de indisponibilidade de terceiros e monitoramento de APIs externas.

4. Equipe e infraestrutura

A equipe é composta por 5 pessoas, todas com experiência em desenvolvimento de software no contexto de grupos de extensão, mercado de trabalho e pesquisa científica. Vamos utilizar ferramentas amplamente difundidas no meio para auxiliar no desenvolvimento do projeto, a exemplo do **VScode, Co-pilot e Github** para fornecer uma melhor experiência de desenvolvimento de software. O discord e o trello serão utilizados para comunicação da equipe e gerenciamento de tarefas respectivamente. Os documentos, atas e outras informações do projeto serão armazenados no **Google Drive.**

Tarefas e responsabilidades iniciais:

1) Cauê (Software Engineer): Investigação de uma potencial integração com o lattes
2) Jesus (Software Engineer): Investigação de outras plataformas para potenciais integrações
3) Letícia (Software Engineer): Investigação de uma potencial integração com o Github e Gitlab
4) Luiz (Product Owner e UX Engineer) : Apoio e investigação da usabilidade e UX da solução e continuação da análise dos requisitos
5) Pedro (Software Engineer): Investigação de uma potencial integração com o google scholar

Próximas Tarefas:

5. Acompanhamento do projeto

Serão realizadas reuniões periódicas no **Discord** entre todos os membros da equipe para dar updates e compartilhar o status das tarefas realizadas por cada um. O **trello** será usado para facilitar o acompanhamento de tasks e o desenvolvimento utilizando o modelo **Kanban**. Reuniões pontuais com os professores entrevistados anteriormente serão realizadas para garantir que o desenvolvimento esteja seguindo os requisitos levantados inicialmente.

Todas as decisões relevantes, mudanças de escopo ou alterações importantes serão documentadas, assegurando histórico claro e rastreável para futuras referências.

6. Cronograma e Marcos do projeto

Milestones (cerca de 1-2 semanas para cada milestone):

Deadline	Goal
25/04	Identificação e análise das plataformas candidatas à integração
02/05	Definição de requisitos, tarefas e responsabilidades para cada integração a serem realizadas
09/05	Início do desenvolvimento das primeiras funcionalidades (provas de conceito de integrações)
23/05	**Entrega do primeiro MVP funcional com filtros de busca e geração básica de currículo**

30/05	Validação do MVP junto aos stakeholders e coleta de feedbacks
06/06	Refinamento do MVP com ajustes baseados nos feedbacks recebidos
13/06	Implementação de funcionalidades complementares, como dashboards de métricas e melhorias na interface
20/06	**Entrega da primeira versão completa do produto**
27/06	Validação final do produto com stakeholders e análise crítica para identificação de melhorias futuras
Pós entrega	Realização de retrospectiva geral do projeto e definição de planos de continuidade e manutenibilidade

Para o primeiro milestone, foi realizada a tarefa de identificação das possíveis formas de integração entre o ORCID e plataformas externas. O levantamento considerou critérios de acessibilidade, disponibilidade oficial de APIs e condições de uso. Foi constatado que o Google Scholar não fornece API pública, o que limita integrações diretas. Alternativas envolvem o uso da biblioteca Python scholarly, baseada em scraping, ou o serviço pago serpAPI, que oferece interface estruturada mediante custo por requisição. No GitHub, identificou-se a existência de uma REST API oficial, pública e amplamente documentada, que permite interações com repositórios, perfis e estatísticas de atividade. O GitLab, por sua vez, disponibiliza a REST API v4 [4], que abrange funcionalidades similares, com suporte nativo a webhooks e tokens de acesso. Em relação ao Lattes, foi confirmado que não há API pública acessível para consultas diretas. A integração oficial ocorre apenas via "Extrator Lattes" [3], destinado a instituições autorizadas e sujeito a processo formal de aprovação, o que pode impor restrições operacionais. Para bases internacionais, foi avaliada a Scopus Search API, da Elsevier, que permite buscas por artigos, autores, afiliações e tópicos de pesquisa, com autenticação via OAuth2 e limites de requisições por chave. Além disso, a Web of Science API Expanded, da Clarivate [5], foi estudada como alternativa, permitindo consultas refinadas sobre artigos indexados, metadados de citações e filtros por áreas temáticas, data de publicação ou fonte. A complexidade de acesso e uso de cada API foi documentada para orientar a fase seguinte de seleção técnica das integrações prioritárias.

7. Gerência de Riscos

A gestão de riscos do projeto visa antecipar possíveis problemas que possam impactar negativamente o cronograma, a qualidade ou a entrega das

funcionalidades previstas. Cada risco identificado possui uma estratégia de mitigação definida para reduzir a probabilidade de ocorrência ou minimizar seus impactos.

Riscos de integração

Após essa primeira investigação, nos deparamos com alguns dos riscos já mencionados na última versão, mas também identificamos novos que podem surgir durante a implementação das integrações.

- Falta de API oficial pública: Plataformas como Lattes e Google Scholar não oferecem APIs públicas para integração direta, o que nos torna dependentes de técnicas de scraping para extração de dados. Para mitigar esse risco, as integrações iniciais serão priorizadas para plataformas que disponham de APIs oficiais. Para plataformas sem suporte oficial, serão avaliadas bibliotecas de scraping ou parsing, respeitando as limitações técnicas.
- Limitação de requisições: APIs públicas impõem limites de chamadas por janela de tempo, enquanto práticas de scraping intensivo podem ocasionar bloqueio de IPs ou de credenciais de acesso. Para mitigar este risco, serão implementadas estratégias de cache de dados para reduzir a quantidade de requisições redundantes, além da utilização de mecanismos de backoff exponencial em tentativas de reconexão. Adicionalmente, será avaliada a utilização de servidores proxy rotativos em cenários controlados, com o objetivo de distribuir o volume de requisições de forma segura e minimizar bloqueios, respeitando os termos de uso de cada plataforma. O monitoramento contínuo dos headers de rate limit retornados pelas APIs será empregado para ajustar dinamicamente a frequência de chamadas.
- Consistência de Dados: As respostas das integrações podem apresentar diferenças de estrutura, incluindo variações no formato (por exemplo, JSON, XML ou CSV), variações de nomenclatura de campos, formatos de data inconsistentes e ausência de informações obrigatórias. Essa heterogeneidade torna o processo de ingestão de dados mais sujeito a erros de interpretação e falhas de integração.

  Para mitigar a inconsistência do processamento, será implementada uma camada de normalização dos dados extraídos, convertendo-os para um modelo interno padronizado, independente da origem. Essa camada terá responsabilidade explícita de tratar campos opcionais, atribuindo valores padrão ou null-safe quando necessário, além de realizar conversões de formato (por exemplo, normalização de datas para padrão ISO 8601).

  Adicionalmente, cada integração será validada contra um esquema de dados previamente definido, utilizando ferramentas de schema validation específicas para o tipo de resposta (por exemplo, JSON Schema para

respostas JSON). Qualquer discrepância encontrada - como campos obrigatórios ausentes, formatos inválidos ou valores inconsistentes - será registrada para auditoria e tratamento posterior.

- Restrições Legais e de Privacidade (LGPD): Regulamentações de privacidade podem exigir consentimento ou limitar uso de dados pessoais. Pensando nisso, vamos extrair apenas dados públicos e consultar o jurídico da instituição antes de publicar funcionalidades.

Riscos durante o processo de desenvolvimento

Durante o processo de desenvolvimento, diversos riscos podem comprometer o cronograma, a qualidade das entregas ou a aderência do sistema às necessidades reais dos usuários. A identificação e o tratamento proativo desses riscos são importantes para minimizar impactos.

Um dos principais riscos é a ocorrência de mudanças frequentes nos requisitos ou no escopo do projeto. Essas alterações podem decorrer tanto de novas demandas vindas dos stakeholders quanto da percepção tardia de funcionalidades que se mostram necessárias apenas após o início da implementação. Mudanças não controladas podem gerar desalinhamento do planejamento original, aumentar o volume de retrabalho e comprometer a entrega de funcionalidades críticas dentro do prazo. Para mitigar esse risco, será implementado um fluxo estruturado de controle de mudanças, com etapas formais de análise de impacto, replanejamento e aprovação antes da incorporação de novas demandas.

Outro risco relevante é a falha em ambientes de desenvolvimento e teste. Instabilidades em servidores, bancos de dados de apoio ou serviços de terceiros integrados podem impedir o avanço das atividades de desenvolvimento e prejudicar a realização de testes de validação. Para reduzir a exposição a esse risco, o projeto manterá ambientes de testes isolados e atualizados, com rotinas de backup e planos de contingência para migração de infraestrutura em caso de falhas prolongadas.

A ocorrência de bugs complexos, que não sejam detectados em testes iniciais, também representa um risco relevante. Problemas de lógica, concorrência ou integração entre componentes podem emergir tardiamente e exigir correções dispendiosas. Para mitigar esse tipo de risco, o projeto adotará uma estratégia de testes robusta, incluindo testes unitários abrangentes, testes de integração entre módulos e testes end-to-end nos fluxos principais da aplicação. O uso de cobertura de código será monitorado para garantir que pontos críticos do sistema estejam efetivamente testados.

Além dos riscos técnicos, existe também o risco de indisponibilidade dos professores e stakeholders entrevistados para realizar sessões de feedback e

validação de funcionalidades. Como certas features do OrSync dependem diretamente da validação prática de usuários finais, a falta de retorno pode atrasar entregas ou forçar suposições sobre o comportamento desejado. Para minimizar este risco, a equipe pretende adotar uma política de comunicação antecipada, agendando entrevistas e checkpoints com antecedência mínima de uma semana, e utilizando canais alternativos de feedback assíncrono (como formulários ou protótipos navegáveis) sempre que possível.

Por fim, mesmo com planejamento e estratégias de mitigação, é reconhecido que nem todos os riscos podem ser completamente eliminados. Por isso, o acompanhamento contínuo dos riscos, revisões semanais de status e a capacidade de adaptação do plano de ação serão fatores chave para assegurar a resiliência do projeto diante de imprevistos.

8. Testes do produto

A qualidade do software será garantida por meio de uma abordagem estruturada de testes, dividida em três níveis principais: testes unitários, testes de integração e testes end-to-end, envolvendo tanto a equipe quanto os stakeholders na validação das funcionalidades.

Testes Unitários: Os testes unitários serão responsáveis por validar o comportamento isolado de funções, métodos e classes críticas do sistema. Cada unidade de código - como componentes de coleta de dados, módulos de normalização, adaptadores de comunicação com APIs externas e módulos de persistência - será testada individualmente para assegurar que produza os resultados esperados. Os casos de teste incluirão variações de entrada válidas e inválidas, simulações de erros de rede, tratamento de dados nulos e validação de formatos.

Testes de integração: Além dos testes unitários, o projeto contará com uma camada de testes de integração, focada na interação entre os diferentes módulos internos do sistema e entre o sistema e serviços externos. Por exemplo, a integração entre o cliente REST de consulta ao ORCID e o banco de dados será testada para verificar se os dados obtidos são corretamente armazenados. Para serviços que requerem autenticação OAuth ou APIs com limites de requisição, serão utilizados mocks e bancos de dados de teste, simulando o comportamento real das plataformas sem afetar o ambiente de produção.

Testes End-to-End (E2E): Os testes end-to-end serão projetados para validar o fluxo completo de interação do usuário final com o OrSync, cobrindo casos que vão desde a configuração inicial do perfil até a sincronização de dados e visualização de relatórios. Inicialmente, esses testes serão realizados pela equipe

de desenvolvimento, utilizando frameworks como Cypress para simulação automatizada de fluxos no frontend. Os testes end-to-end simularão ações típicas dos usuários, como: conectar uma nova plataforma externa, configurar filtros de dados, gerar currículos personalizados e visualizar métricas de citações e coautorias. Casos de erro, como falhas de autenticação ou indisponibilidade de API, também serão testados para assegurar o correto tratamento de exceções.

Validação pelos Stakeholders: Após as rodadas internas de testes, versões candidatas (*release candidates*) serão disponibilizadas para stakeholders selecionados - incluindo professores e pesquisadores que participaram das entrevistas iniciais. Esses usuários validarão o sistema a partir de roteiros de teste previamente preparados pela equipe, com orientações sobre os fluxos a serem testados. O feedback qualitativo (percepção de facilidade, clareza das informações) e quantitativo (detecção de erros ou inconsistências) será registrado e analisado para ajustes antes da liberação final de cada versão.

9. Gerenciamento de configuração de software

O gerenciamento de configuração do projeto será realizado utilizando o GitHub como plataforma de versionamento de código, integrando práticas de controle de issues, organização de branches e revisão de código. O fluxo de trabalho seguirá uma adaptação do Git Flow para melhor adequação ao ciclo ágil de entregas incrementais.

A branch main será destinada exclusivamente a versões estáveis do produto. A branch develop servirá como linha principal de desenvolvimento, integrando as funcionalidades implementadas e validando incrementos antes da promoção para produção. Novas funcionalidades serão desenvolvidas em branches nomeadas no padrão feat/descricao-breve, sempre derivadas de develop, para garantir a organização das implementações. Alterações relacionadas a correções de defeitos detectados durante o ciclo de desenvolvimento utilizarão branches fix/descricao-breve, também originadas a partir de develop. Tarefas ou mudanças que não impactem diretamente as funcionalidades do sistema, como atualizações de dependências, ajustes de configuração ou melhorias de ambiente, serão desenvolvidas em branches chore/descricao-breve. Em casos de necessidade de correções emergenciais diretamente em produção, serão abertas branches hotfix/descricao-breve, derivadas da main, permitindo a rápida resolução de incidentes críticos com impacto mínimo no fluxo de trabalho principal.

Os commits realizados no repositório seguirão um padrão semântico para facilitar a rastreabilidade de alterações e apoiar automações futuras, como

geração de changelogs. O padrão de commit adotado será estruturado no formato tipo(escopo): descrição, conforme descrito na seguinte tabela:

Tipo	Uso
feat	Desenvolvimento de nova funcionalidade
fix	Correção de defeito no código
chore	Tarefas que não impactam funcionalidades

As issues abertas no repositório GitHub serão vinculadas a pull requests relacionados, com descrição objetiva e detalhamento suficiente para rastrear as motivações e os critérios de aceitação da alteração proposta. O processo de pull request exigirá obrigatoriamente uma descrição clara das modificações realizadas, referência à issue correspondente e cumprimento de checklist de critérios técnicos definidos para o projeto.

Durante o ciclo de vida do projeto, as práticas de pull request incluirão a obrigatoriedade de associar cada mudança a uma task correspondente, utilizar mensagens de commit padronizadas [6], seguir os padrões de codificação definidos [7] e documentar eventuais decisões técnicas relevantes. Mudanças que não apresentarem testes automatizados associados, quando aplicável, serão reprovadas até que cumpram os critérios mínimos de qualidade estabelecidos.

Para garantir a integração contínua e a estabilidade das versões entregues, o projeto contará com pipelines de CI/CD configurados através do GitHub Actions. Esses pipelines serão acionados automaticamente a cada push ou pull request enviado para as branches monitoradas (develop e main). Durante o pipeline, serão executadas etapas de instalação de dependências, execução de testes automatizados, validações de estilo de código e geração de artefatos compilados.

Toda vez que alterações forem incorporadas à branch main, o pipeline de CI/CD continuará de forma automática com a etapa de build e entrega contínua. O sistema será configurado para realizar a compilação do projeto e o upload dos artefatos resultantes para um ambiente de cloud computing, que poderá ser configurado em provedores como AWS. Essa configuração visa eliminar processos manuais de deploy e reduzir o tempo entre a conclusão do desenvolvimento e a disponibilização de versões atualizadas para o ambiente de produção.

O processo de deploy será implementado com validações finais pós-build antes da publicação oficial da nova versão. Além disso, mecanismos de rollback serão previstos, possibilitando o retorno imediato para a versão anterior em caso de falhas críticas identificadas após o deploy. Dessa forma, o fluxo de integração e entrega contínua adotado no projeto garantirá maior agilidade no ciclo de releases e maior segurança operacional no processo de atualização do sistema.

10. Plano de Manutenção de software

O OrSync será mantido de acordo com um plano de manutenção baseado em quatro categorias principais: corretiva, adaptativa, evolutiva e preventiva. O objetivo é assegurar que o sistema continue funcionando de forma segura, eficiente e alinhada às necessidades dos usuários após a sua entrega inicial.

A manutenção corretiva será focada na identificação e correção de falhas de funcionamento e vulnerabilidades de segurança. Erros reportados pelos usuários ou identificados nos sistemas de monitoramento serão analisados, priorizados e corrigidos com base no impacto e na urgência, seguindo boas práticas de versionamento e validação em ambiente de staging antes da liberação para produção.

A manutenção adaptativa garantirá que o sistema permaneça compatível com alterações externas, como atualizações de APIs de terceiros, mudanças de autenticação e variações de infraestrutura. Mudanças em ambientes operacionais ou requisitos de integração serão monitoradas e tratadas de forma planejada, assegurando a continuidade dos serviços.

A manutenção evolutiva permitirá a inclusão de melhorias no sistema, tanto no desempenho quanto na usabilidade. Poderão ser adicionadas novas funcionalidades, aprimoramentos nos relatórios analíticos e melhorias nos fluxos de sincronização de dados, baseadas no feedback dos usuários e nas demandas emergentes do contexto acadêmico.

A manutenção preventiva consistirá em ações para preservar a qualidade do código e evitar a degradação do sistema ao longo do tempo. Refatorações, atualização de bibliotecas e ampliação da cobertura de testes automatizados serão realizadas de maneira periódica, reduzindo a ocorrência de falhas futuras e mantendo a base de código estável e segura.

Todas as manutenções seguirão o fluxo de trabalho definido no projeto, com utilização de issues, branches específicas para cada intervenção e revisão técnica antes da integração ao código principal. Dessa forma, o OrSync se manterá atualizado, funcional e preparado para evoluções contínuas.

Referências

[1] https://dev.elsevier.com/documentation/SCOPUSSearchAPI.wadl

[2] https://docs.github.com/en/rest?apiVersion=2022-11-28

[3]https://www.gov.br/pt-br/servicos/obter-acesso-ao-extrator-da-plataforma-lattes

[4] https://docs.gitlab.com/api/rest/

[5] https://developer.clarivate.com/apis/wos

[6] https://cbea.ms/git-commit/ How to write git commit messages

[7]https://testing.googleblog.com/2017/06/code-health-too-many-comments-on-your.html Code Health: Too Many Comments on Your Code Reviews?