Relatório final de atividades

Projeto: "ServiçosPI: inovação e eficiência na recuperação de informações de propriedade intelectual do INPI."

Coordenador do Projeto:	**Washington Luís Ribeiro de Carvalho Segundo**		
Nome do pesquisador:	**Pedro Henrique Cordeiro de Jesus**		
Vigência da Bolsa de Pesquisa:	**01/04/2026**	**Até:**	**30/09/2026**
Encerramento em:	**30/09/2026**		
Ref. Fundação:	**31849**	**Ref. Ibict:**	**0677/2025**

Descrição das Atividades da Bolsa de Pesquisa:

1-Seleção de componentes tecnológicos adequados ao desenvolvimento do módulo de estatísticas do INPI; meta 3. 2-Definição e validação da arquitetura tecnológica, Meta 4. 3- Documento técnico descrevendo a arquitetura tecnológica, incluindo requisitos de hardware e software, meta 4. 4- Relatório das atividades desenvolvidas

1 INTRODUÇÃO

O presente relatório documenta as atividades desenvolvidas no período de 1º de abril de 2026 a 30 de setembro de 2026, como bolsista do Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT), no âmbito do projeto "ServiçosPI: inovação e eficiência na recuperação de informações de propriedade intelectual do INPI". Encerrada a etapa anterior — de deployment e configuração da infraestrutura de dados em Docker Swarm (Apache Airflow e MinIO) —, o período aqui descrito concentrou-se na construção do pipeline de dados propriamente dito: na tradução das regras de negócio do INPI em código e a montagem, camada por camada, da esteira que leva um pedido de patente do banco Informix legado até um dataset gold validado e enriquecido com o andamento processual.

O ativo escolhido para essa primeira esteira completa foi o de Patentes, dado seu volume e a maturidade das regras de negócio já mapeadas pelo INPI. O pipeline resultante — camada silver de ingestão incremental, camada gold de aplicação das regras de negócio e uma camada de enriquecimento de status via integração com o BPMS SYDLE ONE do INPI — foi deliberadamente desenhado para ser reaproveitado: a partir de agosto de 2026, a estrutura do repositório, o padrão de deploy e o desenho das DAGs de patentes tornaram-se o template de referência para os próximos ativos de dados do projeto (marcas, desenhos industriais e programas de computador).

Sustentar esse pipeline exigiu também expandir a infraestrutura de processamento além do que havia sido implantado no ciclo anterior: a instalação e o dimensionamento do Apache Spark em modo cluster no Docker Swarm, a disponibilização de um servidor Spark Connect para que as DAGs do Airflow submetam jobs sem embutir uma JVM, e adoção de NFS como armazenamento POSIX compartilhado entre os nós do cluster. O resultado foi validado ponta a ponta contra um gabarito oficial de pedidos fornecido pelo INPI. Este relatório está estruturado em Método, que descreve como cada componente foi construído, Resultados, que apresenta o que foi entregue, validado e as limitações em aberto, e Referências.

2 MÉTODO

As atividades do período seguiram a ordem em que os componentes do pipeline se tornam dependentes entre si: primeiro, a reorganização dos repositórios e o deploy que os sustenta, depois, a infraestrutura de processamento compartilhada, e, por fim, as três DAGs de dados propriamente ditas.

2.1 Organização de Repositórios e Padrão de Deploy

Até julho de 2026, todas as DAGs do projeto viviam em um único repositório, servicospi-data-ingestion, cujo Jenkinsfile monolítico (504 linhas, oito estágios) construía imagens, publicava o stack Swarm completo e sincronizava DAGs numa única pipeline. Esse acoplamento tinha custo visível: de abril a maio, 46 dos cerca de 594 commits do repositório trataram exclusivamente do diagnóstico de rede e do healthcheck do próprio Jenkinsfile — sintoma de um deploy pesado, em que qualquer alteração exigia tocar o mesmo pipeline. Em 17 de agosto, criei o repositório dedicado silver, já com a DAG de ingestão isolada; em 31 de agosto, seu Jenkinsfile foi reduzido a 136 linhas e dois estágios (Setup e Deploy), publicando as DAGs em um subdiretório próprio no NFS. No dia seguinte, repliquei o mesmo padrão para o repositório patentes (DAG gold_patentes), e em 1º de setembro, as DAGs legadas equivalentes foram removidas do repositório original. O par silver/patentes consolidou o template — repositório dedicado, pasta dags/, Jenkinsfile de dois estágios e README — reservado para os próximos ativos de dados, enquanto o servicospi-data-ingestion passou a reter apenas a infraestrutura compartilhada e a ingestão bronze via Logstash.

2.2 Infraestrutura de Processamento: Spark, Spark Connect e NFS

A instalação do Apache Spark começou em maio, com uma imagem própria (services/spark) construída sobre apache/spark:3.5.0-python3, com Python 3.12 compilado do zero para casar com a versão do driver Airflow, e os jars de Iceberg (1.6.1), Delta (3.2) e Hadoop-AWS já embutidos. O cluster roda em modo Swarm com três

workers (12 GB de heap, 4 CPUs cada); os nós reais variam de 8 CPUs/15 GB em homologação a 16 CPUs/125 GB em produção.

A partir de 5 de julho, subi um serviço Spark Connect dedicado (6 GB de driver e de executor): um servidor gRPC com driver persistente que concentra os catálogos Iceberg, permitindo que as DAGs do Airflow atuem como clientes finos (pyspark[connect], sem JVM local). O ganho ficou claro na operação diária: como o driver do Spark Connect fica sempre ativo, cada task do Airflow apenas abre uma sessão remota contra um cluster já quente, em vez de pagar o custo de subir uma aplicação Spark nova a cada execução — silver, gold e enriquecimento passaram a compartilhar o mesmo cluster warm, o que tornou a esteira sensivelmente mais fluida entre uma camada e outra, além de reduzir o custo por task do Celery e o acoplamento entre o worker do Airflow e o cluster de processamento. Ao longo do período, registrei três catálogos Iceberg distintos nesse servidor: iceberg (MinIO, dados históricos), iceberg_nfs (silver do Informix) e, em 1º de setembro, iceberg_silver (status de pedidos via SYDLE ONE).

Os dois catálogos mais novos escrevem diretamente no NFS (warehouse em file://, via HadoopCatalog), em vez de passar pela API S3 do MinIO, como o catálogo histórico. Essa mudança removeu um gargalo real de I/O: cada commit do Iceberg no MinIO depende de chamadas HTTP da API S3 para os arquivos de metadado e manifests, o que penaliza justamente o padrão de escrita frequente e em lote pequeno do merge-on-read; escrevendo direto no filesystem POSIX do NFS, esse custo de rede desaparece e o merge passa a ser limitado pelo disco compartilhado, não pela API de objetos. O mesmo volume NFS (parametrizado por ${NFS_DATA_PATH}, ${DOCKER_HOST_IP}, ${PROJECT} e ${ENVIRONMENT}) também serve como filesystem POSIX comum ao Airflow (DAGs, logs, CeleryExecutor) e ao Logstash (staging da camada bronze), independentemente de qual nó físico do Swarm execute cada container.

2.3 Camada Silver: Ingestão Incremental para Iceberg

A DAG prod_silver_informix, consolidada em 1º de setembro, é o primeiro elo da esteira: ingere as 29 tabelas (8 crp_* e 21 ptn_*) que o Logstash já exportou do Informix para NDJSON na camada bronze do NFS, aplicando, tabela a tabela, deduplicação por chave primária e um único MERGE INTO no Iceberg por tabela/dia, com retentativa automática em caso de conflito de commit. As nove maiores tabelas usam hidden partitioning nativo do Iceberg (bucket sobre a mesma coluna da cláusula ON do merge), garantindo partition pruning sem exigir coluna de partição explícita no schema. A idempotência é garantida por marcadores de sucesso/falha no NFS, gravados somente após o commit do merge, e o catálogo iceberg_nfs desta DAG é isolado do catálogo em MinIO usado pela ingestão histórica, evitando interferência entre as duas cargas. Ao final da ingestão das 29 tabelas, um TriggerDagRunOperator dispara a DAG prod_gold_patentes sem esperar sua conclusão — a gold lê direto as tabelas silver recém-atualizadas via Spark Connect, monta um rolling_df a partir de ptn_pedido (filtrado por natureza e pelo lote incremental) e aplica, uma a uma, as regras de negócio detalhadas na Seção 2.4. É esta a DAG-template citada na Seção 2.1.

2.4 Camada Gold: Tradução das Regras de Negócio de Patentes em Código

Das atividades do período, a tradução das regras de negócio de patentes do INPI para código, foi a que exigiu mais tempo e mais cuidado: são dezoito regras de negócio oficiais, implementadas em dezenove funções — regra_00 a regra_18, já que duas seções (kind code e PCT) são calculadas em duas etapas cada — aplicadas sequencialmente como LEFT JOIN sobre um rolling_df que começa nos pedidos elegíveis (ptn_pedido filtrado por natureza e pelo lote incremental) e termina no registro gold, uma linha por cd_pedido, sem nunca perder ou reordenar uma linha no caminho. Cada regra tem uma seção normativa dedicada, com INID, fonte e base legal; a tabela a seguir resume cada uma, na ordem real de execução do pipeline:

Quadro 1: (Identificar)

Regra	Como foi implementada
regra_00 — Base (§2.1, §6, §10, §12)	Direto de ptn_pedido, sem JOIN: calcula as variações do número do pedido (dígito verificador módulo 11, máscara, forma normalizada), data de depósito, título (normalizado para NULL quando inválido) e natureza (derivada dos 2 primeiros caracteres do no_pedido).
regra_01 — Depositantes (§14)	LEFT JOIN ptn_requere_pedido + crp_pfpj, ordenado por no_ordem; gera applicants (completa, com CPF/CNPJ) e applicants_redacted (sem CPF/CNPJ, usada em pedidos sigilosos).
regra_02 — Inventores (§15)	LEFT JOIN ptn_invento_pedido + crp_pfpj, excluindo inventores marcados como anônimos (cd_anonima = 'S') antes do JOIN.
regra_03 — Procuradores (§16)	LEFT JOIN ptn_procura_pedido + crp_pfpj, ordenado por dt_inicio; remove por regex sufixos históricos do nome, ex. "(NOME ANTERIOR...)", antes de normalizar.
regra_04 — Classificação IPC (§13.2)	JOIN ptn_classif_pedido + ptn_classif; array ordenado por no_ordem_pedido — a primeira entrada é a classificação principal.
regra_05 — Classificação CPC (§13.3)	JOIN ptn_classif_pedido_cpc + ptn_classif_cpc, restrito a pedidos PI e MU (cd_sistema_classif válido).
regra_06 — Prioridades unionistas (§18)	JOIN ptn_priorid_pedido + crp_pais, ordenado pela data do depósito de origem; país sem correspondência mantém country_name em NULL.
regra_07 — Despachos (§19)	JOIN ptn_despacho + ptn_tipo_despacho + crp_programa_rpi; só despachos não anulados, ordem cronológica; datas-sentinela do Informix (< 1901) tratadas como NULL.
regra_08 — Flags de kind code (§2.2, etapa 1/2)	Camada intermediária: deriva 5 flags booleanas de evento (publicação, retificação, concessão) a partir dos despachos; o resultado fica em cache para ser reconsumido pela regra_17, mais adiante na esteira.
regra_09 — Renumeração (§5)	JOIN ptn_renumeracao; histórico de números anteriores do pedido, do mais recente ao mais antigo (migrações de formato, ex. 7 para 12 dígitos em 1997).
regra_10 — Vínculos entre	JOIN ptn_vinculo_pedido + ptn_pedido; classifica divisão, adição e prioridade interna em 5 listas por papel (origem/derivado) a

pedidos (§9)	partir do mesmo vínculo bruto.
regra_11 — Data de publicação (§7)	MAX(data) entre os despachos de publicação válidos (1.3, 3.1, 3.2, 3.6, 3.9) não anulados — um pedido pode ser republicado.
regra_12 — Data de concessão (§8)	Despacho 16.1 (ou 23.9/23.12 para PP); usa MAX(data) quando há mais de um despacho de concessão para o mesmo pedido.
regra_13 — Data de fase nacional PCT (§17, proxy)	Aproximada pelo despacho 1.3, já que o campo oficial (via crp_protocolo) não está disponível na ingestão silver.
regra_14 — Resumo e descrição técnica (§11)	JOIN ptn_texto_pedido, com deduplicação por evento ativo e dt_cadastro mais recente — a regra citada mais abaixo pelo custo de carregar o texto integral do abstract.
regra_15 — Informações do PCT (§17)	Número/data do pedido internacional e número/data de publicação OMPI, via JOIN com ptn_pct; pct_number sem wo_number alimenta a regra de sigilo (§2 do RN141).
regra_16 — Situação do pedido (§3)	Implementada e testada, mas não chamada em produção desde 18/08/2026 — a fonte oficial de status migrou para o BPMS (SYDLE ONE); status e status_metadata ficam fora do gold até a regra ser religada a essa fonte.
regra_17 — Kind code final (§2.2, etapa 2/2)	Combina as flags da regra_08 com a natureza numa tabela de prioridade (concessão+retificação > concessão > publicação+retificação > publicação); recalcula também o número de publicação.
regra_18 — Sigilo, RN141	Avalia 4 condições legais encadeadas (PCT não publicado, regras de publicação pré/pós-1997, regras específicas para pedidos divididos) e mascara ~20 campos dos pedidos sigilosos, preservando applicants_redacted e divided_parent.

Fonte?

Essa tradução foi validada contra um gabarito oficial de 319 pedidos fornecido pelo INPI, o que revelou um bug real de deduplicação de depositantes — 16 dos 289 pedidos comparados traziam simultaneamente a razão social antiga e a nova do mesmo CNPJ, pois a troca de razão social gera um novo identificador interno sem encerrar o vínculo antigo — corrigido ao replicar na regra_01 o padrão já usado na regra_03. A mesma validação catalogou divergências que não eram bugs (despachos anulados, kind

codes retificados, títulos desatualizados na base de comparação) e um caso ainda aberto, de truncamento do resumo de um pedido específico. Em setembro, reescrevi o processamento de snapshot completo para incremental, com watermark por tabela sobre as mesmas colunas que o Logstash já usa na camada bronze — a regra_14, que carrega o texto integral do abstract, tornava o reprocessamento diário do snapshot completo caro demais para manter.

2.5 Enriquecimento de Status via Integração com o SYDLE ONE

A terceira DAG da esteira enriquece cada pedido de patente com seu andamento processual, obtido do BPMS SYDLE ONE do INPI. A DAG incremental prod_silver_status_pedido consulta a API do SYDLE ONE por período, com paginação e autenticação por token armazenado como Variable do Airflow, grava um watermark no NFS (usando o próprio Spark para leitura/escrita, por compatibilidade de permissão entre os usuários do Spark e do Airflow) e aplica um MERGE INTO na tabela iceberg_silver.status_pedido, particionada por mês. Para a carga histórica, prod_silver_status_pedido_backfill resolve o intervalo completo a partir da data de depósito mais antiga, fatia-o em janelas de quinze dias — respeitando o limite de 31 dias da API —, faz staging em disco no NFS e sobrescreve a tabela com INSERT OVERWRITE em modo de partição estática. Uma terceira DAG, teste_conexao_sydle_one, valida a conectividade com o ambiente de homologação do INPI, utilizando com uma credencial própria, isolada da de produção. As três DAGs foram entregues e testadas em 1º de setembro de 2026.

2.6 Validação dos Dados: Comparação com o Gabarito do INPI

A validação da camada gold foi feita por comparação direta contra um gabarito oficial de 319 pedidos de patente fornecido pelo INPI, em formato CSV. Um script próprio carrega o gabarito e a saída gold em Iceberg, normaliza os dois lados (nomes de coluna, encoding, formatação de data) e produz, pedido a pedido, uma tabela de divergências, separando bugs reais do pipeline de discrepâncias de origem na própria

base de comparação. Foi essa comparação, sobre os 289 pedidos com depositante preenchido, que expôs o bug de deduplicação corrigido na Seção 2.4.

3 RESULTADOS

Os resultados do período se dividem entre o que foi entregue e validado, as limitações que permanecem em aberto e as contribuições que ficam como base para os próximos ativos.

3.1 Pipeline de Patentes Validado como Template

Ao final do período, as três camadas — silver, gold e enriquecimento de status — estão em produção e encadeadas: a ingestão silver dispara a gold ao final de cada execução diária, e a regra de situação do pedido (regra_16) na camada gold já lê o status enriquecido pela integração com o SYDLE ONE, apoiadas pela mesma infraestrutura compartilhada de Spark Connect e NFS. A tradução das regras de negócio foi validada contra o gabarito oficial do INPI, com o bug de deduplicação de depositantes identificado e corrigido antes de qualquer consumo posterior dos dados. Tão importante quanto o resultado funcional é o padrão que ele deixou: repositório dedicado por ativo, Jenkinsfile de dois estágios, DAGs desenhadas em torno de watermarks e merge-on-read no Iceberg — replicado do silver para o patentes em um único dia de intervalo, e já citado explicitamente como referência nos commits que o seguiram. É esse padrão, mais do que qualquer DAG individual, que reduz o esforço esperado para os próximos ativos do projeto (marcas, desenhos industriais, programas de computador).

3.2 Limitações e Pontos Abertos

Um ponto específico permanece aberto: durante a validação contra o gabarito, foi identificado um pedido cujo resumo aparece truncado na camada gold, com hipótese de falha na reextração incremental do Logstash ainda não confirmada.

3.3 Contribuições e Próximos Passos

A principal contribuição do período é o template de pipeline de dados — arquitetura de repositório, padrão de deploy e desenho de DAG — validado em produção com um ativo real e pronto para ser replicado. Somem-se a isso a infraestrutura de processamento (Spark, Spark Connect e NFS) dimensionada e compartilhada entre as três camadas, e as regras de negócio do INPI traduzidas para código de forma rastreável, regra a regra. Como próximos passos, recomendo: (i) investigar e encerrar a anomalia de truncamento de resumo identificada na validação contra o gabarito; e (ii) replicar, no próximo ativo de dados do projeto, o template de repositório, deploy e DAGs consolidado nesta bolsa.

REFERÊNCIAS

APACHE SOFTWARE FOUNDATION. Apache Spark Documentation. Versão 3.5.0, 2025. Disponível em: https://spark.apache.org/docs/3.5.0/. Acesso em: 20 mai. 2026.

APACHE SOFTWARE FOUNDATION. Spark Connect Overview. 2025. Disponível em: https://spark.apache.org/docs/3.5.0/spark-connect-overview.html. Acesso em: 05 jul. 2026.

APACHE ICEBERG. Iceberg Table Spec e documentação. Versão 1.6, 2025. Disponível em: https://iceberg.apache.org/docs/1.6.1/. Acesso em: 17 ago. 2026.

APACHE SOFTWARE FOUNDATION. Apache Airflow Documentation. Versão 2.11.2, 2025. Disponível em: https://airflow.apache.org/docs/apache-airflow/2.11.2/. Acesso em: 01 set. 2026.

DATABRICKS. The Databricks Lakehouse Platform: Medallion Architecture. 2024. Disponível em: https://www.databricks.com/glossary/medallion-architecture. Acesso em: 17 ago. 2026.

IETF. RFC 7530: Network File System (NFS) Version 4 Protocol. 2015. Disponível em: https://www.rfc-editor.org/rfc/rfc7530. Acesso em: 03 mai. 2026.

BRASIL. Instituto Nacional da Propriedade Industrial. REGRAS.md: especificação de regras de negócio para exportação de dados de patentes. Rio de Janeiro: INPI, 2026. Documento interno.

BRASIL. Instituto Nacional da Propriedade Industrial. Gabarito de validação de pedidos de patente. Rio de Janeiro: INPI, 2026. Documento interno.

BRASIL. Instituto Brasileiro de Informação em Ciência e Tecnologia. Projeto ServiçosPI: integração com o BPMS SYDLE ONE para status de pedidos. Brasília: IBICT, 2026. Documento interno.

gov.br
Documento assinado digitalmente
PEDRO HENRIQUE CORDEIRO DE JESUS
Data: 06/10/2026 23:32:20-0300
Verifique em https://validar.iti.gov.br

Pedro Henrique Cordeiro de Jesus
Pesquisador

De acordo:

gov.br
Documento assinado digitalmente
WASHINGTON LUIS RIBEIRO DE CARVALHO SEG
Data: 08/10/2026 06:06:45-0300
Verifique em https://validar.iti.gov.br

Washington Luís Ribeiro de Carvalho Segundo
Coordenador do projeto