Graduação em Engenharia da Computação
PROPOSTA DE TRABALHO DE CONCLUSÃO DE CURSO

Nome completo do(a) aluno(a): Vinícius Cardoso Novaes
Curso: Engenharia da Computação
E-mail do(a) aluno(a): vcn2@cin.ufpe.br
Título do trabalho: Toccata: Uma implementação baseada em TCC para suporte de transações distribuídas em Elixir
Área: Sistemas Distribuídos; Linguagens de Programação
Nome do orientador: André Luís de Medeiros Santos
Possíveis avaliadores (dois): Carlos André Guimarães Ferraz, Breno Alexandro Ferreira de Miranda

Resumo da proposta:

Transações distribuídas são operações que ocorrem em ambientes distribuídos, envolvendo múltiplos nós conectados em rede e, portanto, não se limitam apenas a bancos de dados. O princípio fundamental desse tipo de transação é o de consistência global - ou todos os nós executam com sucesso, ou nenhum deles executa, garantindo que o sistema permaneça em um estado válido [1][2].

Para alcançar esse objetivo, diferentes padrões de coordenação de transações têm sido propostos. Um dos mais utilizados é o padrão Saga [3], no qual cada operação é executada de forma independente e, em caso de falha, ações compensatórias são realizadas para desfazer os efeitos das etapas anteriores. Outra abordagem é o padrão Try-Cancel/Confirm (TCC) [4], que estrutura cada operação em três fases — tentativa, confirmação e cancelamento — permitindo reservar recursos antecipadamente e realizar confirmações explícitas. Enquanto o padrão Saga prioriza a compensação eventual após falhas, o TCC proporciona maior controle sobre a consistência e reduz o risco de inconsistências temporárias em sistemas distribuídos.

No ecossistema Elixir, linguagem funcional criada por José Valim e executada sobre a máquina virtual Erlang (BEAM) [5], processos são isolados e comunicam-se exclusivamente por mensagens, o que elimina condições de corrida e torna o modelo de concorrência seguro por construção. Essa arquitetura permite a construção de sistemas altamente

distribuídos, tolerantes a falhas e escaláveis, como WhatsApp, CouchDB e RabbitMQ [6]. Mesmo a concorrência do Elixir não compartilhar recursos por padrão, é necessário que haja coordenação para que haja consistência.

Apesar dessas vantagens, a comunidade Elixir ainda enfrenta desafios em relação à integração e à disponibilidade de bibliotecas open-source, conforme aponta a pesquisa *State of Elixir 2024,* realizada pela Curiosum [7]. No contexto de transações distribuídas, a principal biblioteca existente é a Sage [8] , uma implementação do padrão Saga. Contudo, não há implementações conhecidas do padrão TCC nesse ecossistema, nem escrita em Elixir, nem em Erlang. Em contraste, outras plataformas — como o ecossistema Java — já contam com diversas soluções consolidadas, como Apache Seata [9], ByteTCC [10] e Atomikos TCC [11], que suportam múltiplos padrões de transações distribuídas.

Diante desse cenário, o objetivo deste trabalho é propor e desenvolver uma biblioteca open-source para Elixir que implemente o padrão Try-Cancel/Confirm (TCC), contribuindo para o fortalecimento do ecossistema BEAM e oferecendo aos desenvolvedores uma alternativa robusta à Sage.

A avaliação do projeto será conduzida por meio de uma análise comparativa de desempenho e funcionalidades, entre a nova biblioteca e a Sage, considerando métricas como tempo de rollback, comparação de complexidade cognitiva (configuração e quantidade de linhas de código) e tempo de execução completa da transação, avaliando a viabilidade e o desempenho da abordagem TCC em comparação com o modelo de saga no ecossistema BEAM. Para realizar essa comparação, buscaremos procuraremos um projeto open source que utilize a biblioteca Sage, caso não encontremos, criaremos um *toy project* que será utilizado para realizar os experimentos.

Recife, 07 de outubro de 2025

Referências:

[1] Jim Gray and Andreas Reuter. 1992. Transaction Processing: Concepts and Techniques (1st. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[2] M. van Steen and A.S. Tanenbaum, Distributed Systems, 4th ed., distributed-systems.net, 2023.

[3] Hector Garcia-Molina and Kenneth Salem. 1987. Sagas. In Proceedings of the 1987 ACM SIGMOD international conference on Management of data (SIGMOD '87). Association for Computing Machinery, New York, NY, USA, 249–259. https://doi.org/10.1145/38713.38742

[4] Pardon, Guy. "Business Transactions, Compensation and the Try-Cancel/Confirm (TCC) Approach for Web Services." _Business Transactions, Compensation and the Try-Cancel/Confirm (TCC) Approach for Web Services_.

[5] DA SILVA ALMEIDA, Raul André; JUNIOR, Udo Fritzke. Desenvolvimento de Sistemas Paralelos e Distribuídos em Elixir.

[6] ELIXIR. *Case studies*. Disponível em: https://elixir-lang.org/cases.html. Acesso em: 17 out. 2025.

[7] Curiosum, “State of Elixir 2024 Survey,” _Elixir Hub_, 2024. [Online]. Available: [https://elixir-hub.com/surveys/2024](https://elixir-hub.com/surveys/2024?utm_source=chatgpt.com)

[8] HEXDOCS. *Sage — README*. Disponível em: https://hexdocs.pm/sage/readme.html. Acesso em: 17 out. 2025.

[9] APACHE SEATA. *Apache Seata — The Apache Software Foundation*. Disponível em: https://seata.apache.org. Acesso em: 17 out. 2025.

[10] LIUYANGMING. *ByteTCC*. Disponível em: https://github.com/liuyangming/ByteTCC. Acesso em: 17 out. 2025.

[11] ATOMIKOS. *Atomikos — WebHome*. Disponível em: https://www.atomikos.com/Main/WebHome. Acesso em: 17 out. 2025.