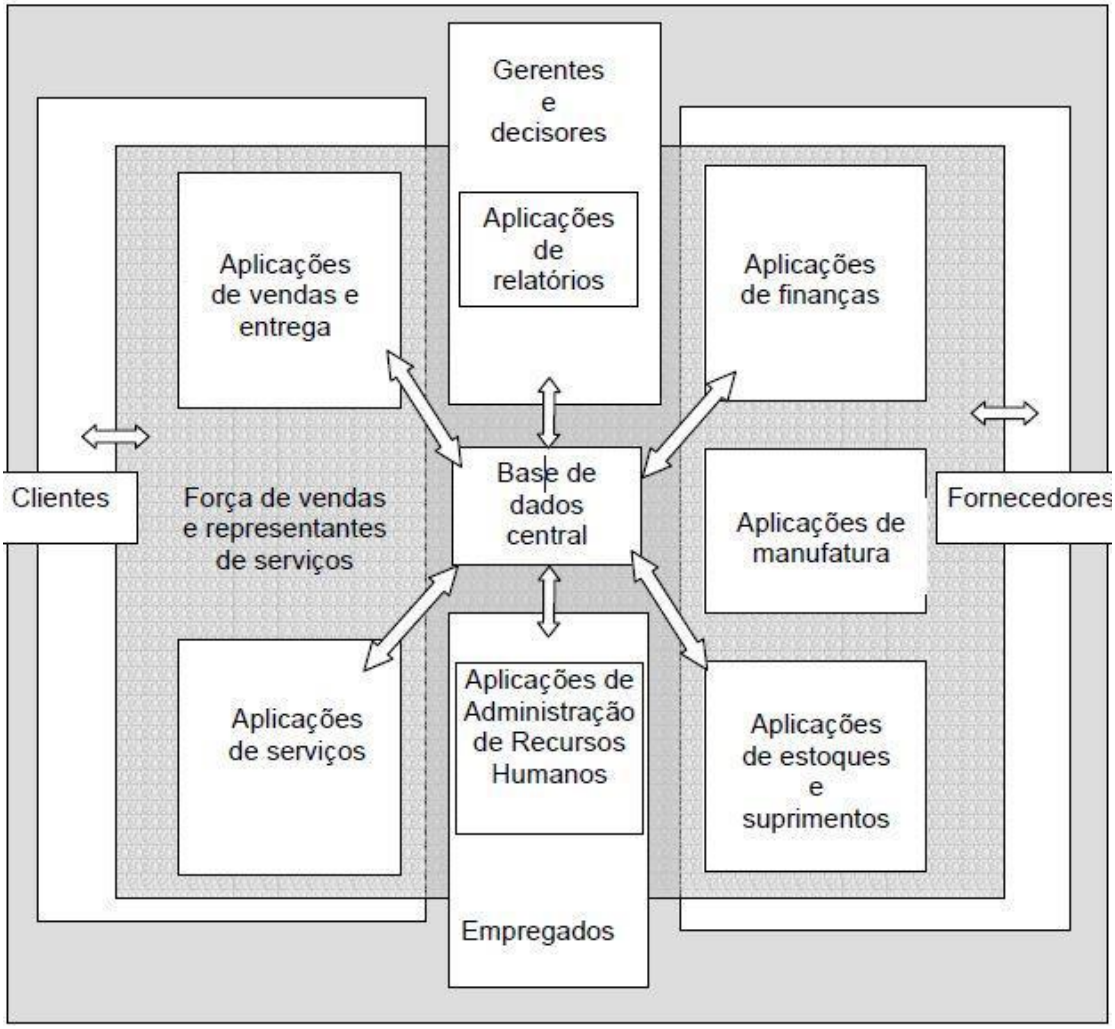
Nome: Davi Augusto Neves Leite

RA: 191027383

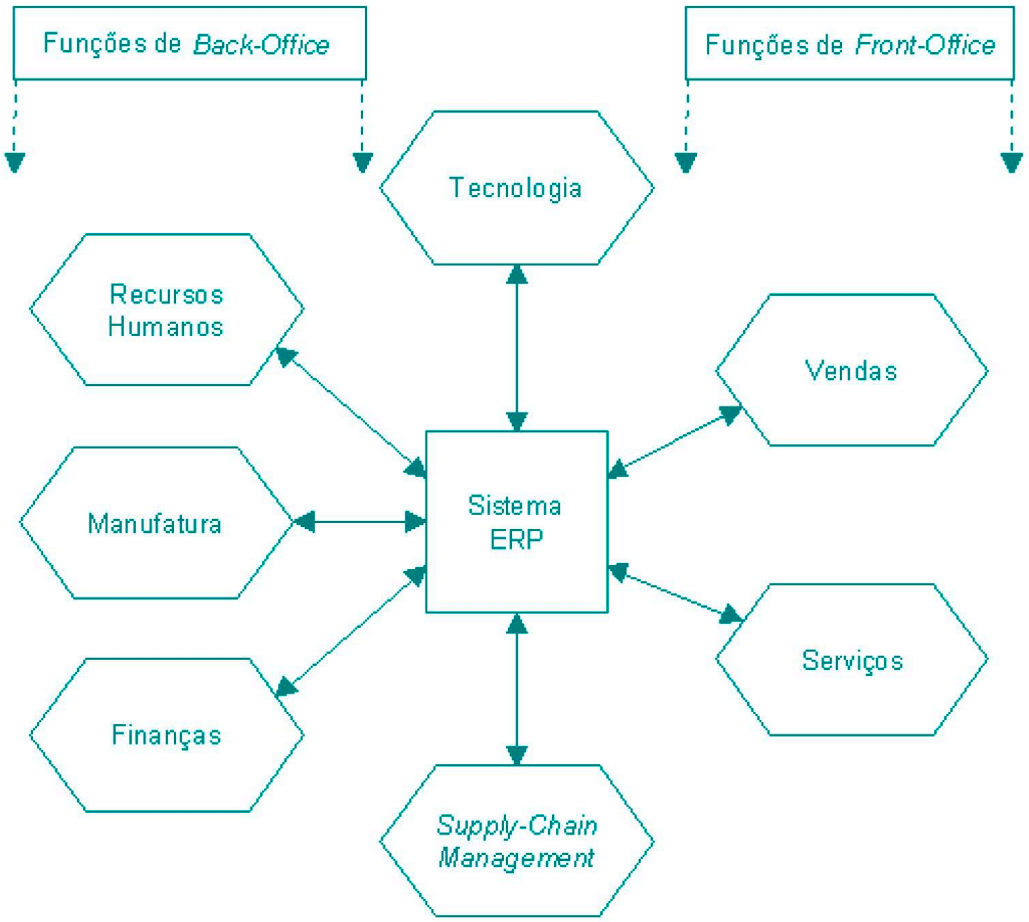
Atividade 4 – Projeto Arquitetural

1) Um sistema ERP, de acordo com Davenport (1998) e Padilha e Marins (2005), significa um “Planejamento dos Recursos da Empresa” que, em outras palavras, simboliza um fornecimento ao controle e suporte a todos os processos operacionais, produtivos, administrativos e comerciais de uma empresa. Dessa forma, diz-se que o ERP é um sistema integrado que permite o fluxo de informações único, continuo e consistente por toda a empresa, sob uma única base de dados.

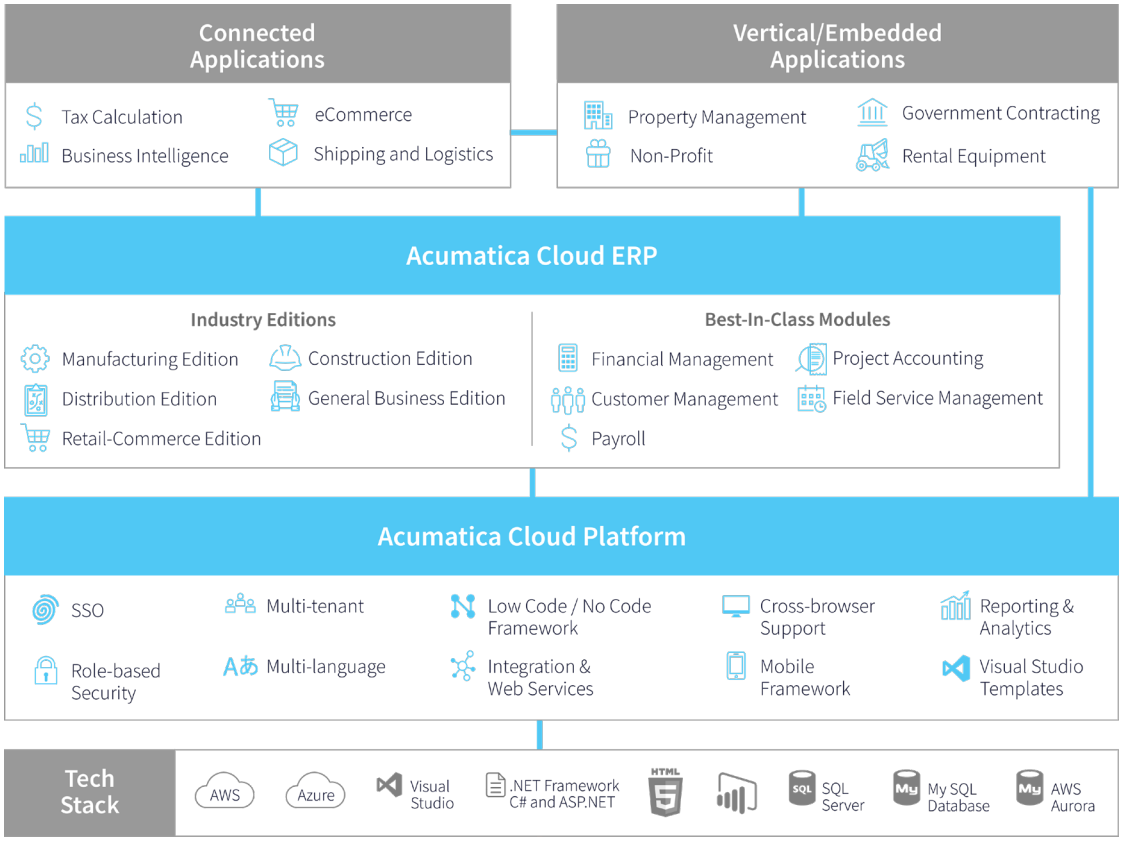
Davenport (1998) trouxe a seguinte **arquitetura conceitual** a respeito de um ERP geral (utilizado até os dias de hoje):



Um outro esquema, mais resumido, é dado por Padilha e Marins (2005) em:



Como exemplo para pequenas e médias empresas, existe um ERP em nuvem denominado ***Acumatica***. Este ERP possui uma arquitetura baseada em camadas, para maior proteção e facilidade no gerenciamento de dados, conforme a empresa retrata na seguinte imagem:



Referências:

https://diceus.com/types-of-erp-systems/

https://www.acumatica.com/acumatica-cloud-xrp-platform/

http://facweb.cs.depaul.edu/jnowotarski/is425/hbr%20enterprise%20systems%20davenport%201998%20jul-aug.pdf (Davenport)

https://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-65132005000100009 (Sistemas ERP: características, custos e tendências)

2)

A) Um radar animado, bem como a maioria das tecnologias associadas a meteorologia, necessita de um sistema baseado em atualizações praticamente em tempo real para o monitoramento. Dessa forma, a arquitetura principal que mais se adequa a esse tipo de aplicação é a de **repositório**, tendo a peça central como sendo a principal máquina que recebe as informações advindas dos sensores metrológicos. Além disso, como essa aplicação está funcionando

em uma **página da *internet***, aplica-se a arquitetura de **cliente-servidor** associada as requisições feitas pelos usuários locais para o servidor associado a essa página (https://www.ipmetradar.com.br/2animRadar.php).

Referências:

https://www.ipmetradar.com.br/2animRadar.php

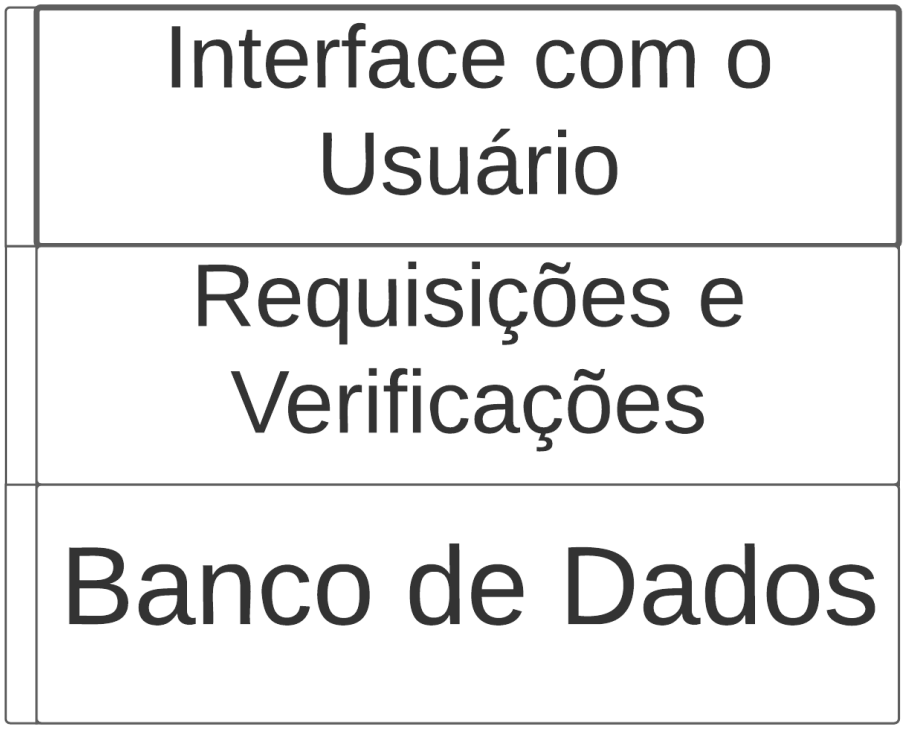
B) Um software de gestão de entregas, como o *Find MyPack*, está atrelado a verificações constantes do estado dos itens listados. Ou seja, o software realiza periodicamente atualizações em seu sistema, com base no estado atual do objeto. Dessa forma, a arquitetura mais predominante nesse sistema é a **MVC**, consistida no modelo (objetos para rastreamento), visualizador (interação com usuário e requisições) e controlador (atualização periódica).

Referências:

http://www.controlog.net.br/home

https://www.findmypack.com.br/

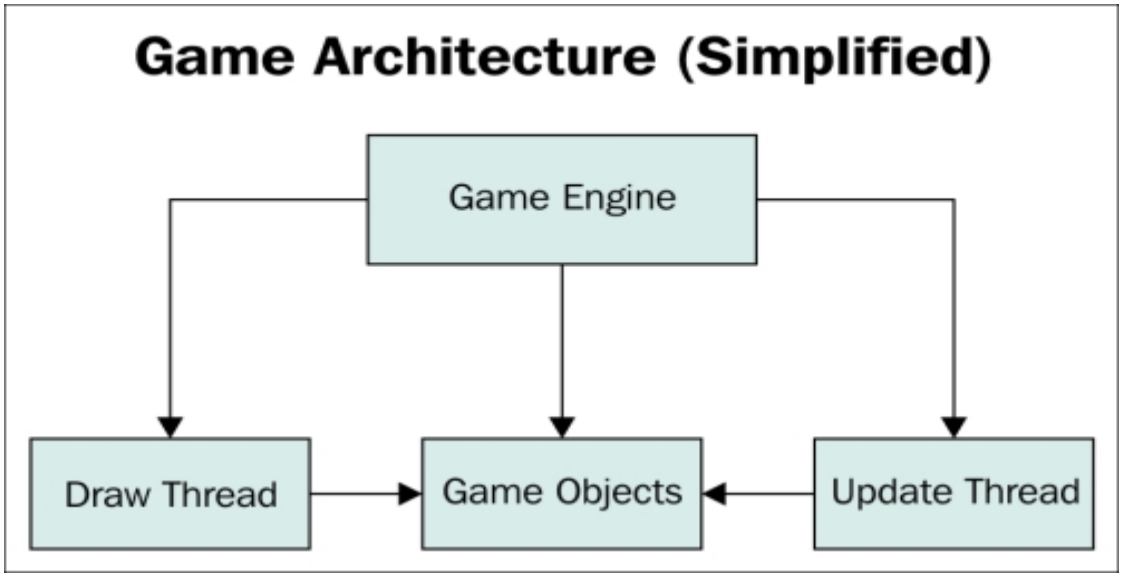
C) Um sistema de caixa eletrônico de um banco está associado a arquitetura em **camadas**, uma vez que as funções são bem divididas e a comunicação ocorre de camada a camada. Por exemplo: o cliente solicita uma requisição, como acessar sua conta corrente. A partir dessa requisição, é enviado uma solicitação ao banco de dados associado, o qual devolve as informações que são processadas pelo *software*. Internamente a isso, existe a arquitetura de **cliente-servidor**, uma vez que os dados estão armazenados em um servidor e são feitas as requisições pelo cliente (caixa eletrônico)**.** Isso pode ser visto no desenho esquemático abaixo e é o que confere a maior “lentidão” e segurança dos *softwares* de caixa eletrônico:



Referências:

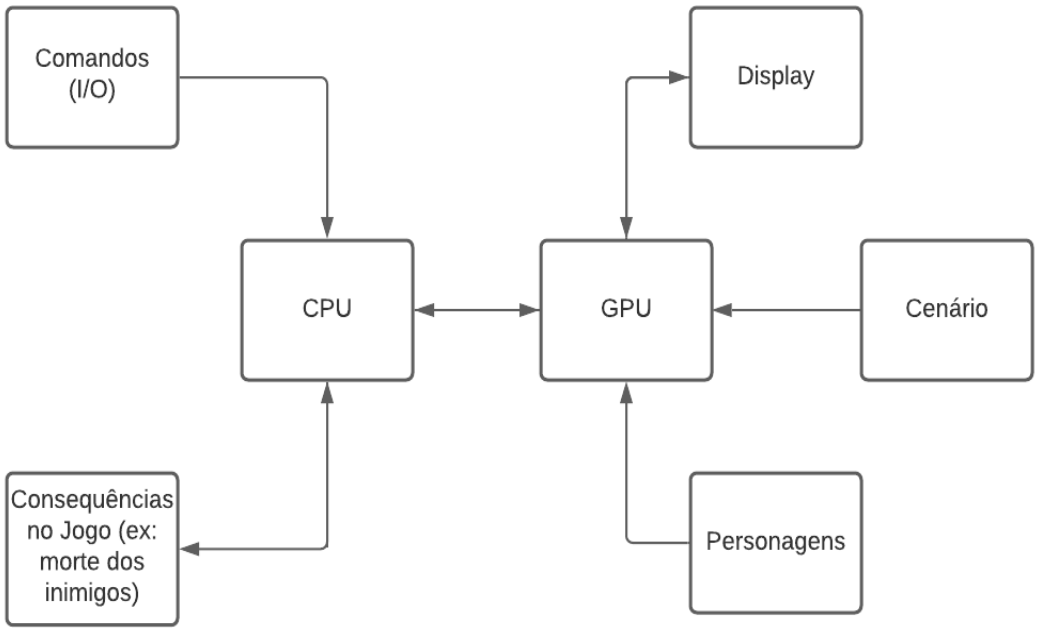
https://www.ic.unicamp.br/~ariadne/mc436/2s2010/cap6.pdf

D) Um jogo eletrônico, por si só, envolve diversos componentes funcionando **simultaneamente**, como renderização de cenários e armas, processamento de comandos (a fim de se garantir a interatividade), dentre outros. Além disso, tem-se a questão de **recursão**, haja visto que um jogo só pode ser encerrado em decorrência de algum problema interno (famosos *crash's*) ou a pedido do usuário. Um esquema simplificado da arquitetura de um jogo feito em um motor específico (como Unity) pode ser visto abaixo:

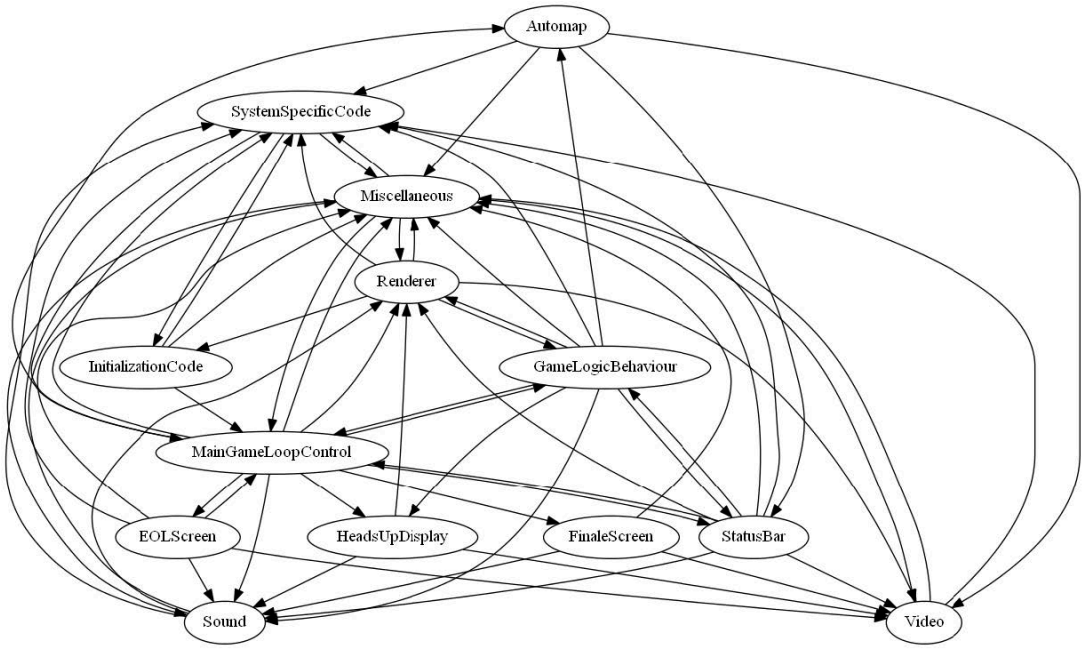


Dessa forma, um jogo do tipo first person shooter (FPS) como Call of Duty, Overwatch e Battlefield, é necessária uma gama de componentes interligados entre si de modo a trazer a experiência correta ao usuário, pois há uma grande renderização de cenários e armas somado a interatividade (comandos como pular, mover, atirar, dentre outros). Dessa forma, a arquitetura mais recomendada para esse tipo de aplicação é a de **repositório**, pois os componentes são interligados numa peça central (essa que seria a CPU e GPU).

Um esquema pode ser demonstrado abaixo:



Além disso, de acordo com Prashar (2014), tem-se um exemplo do subsistema de renderização de cenário do jogo Doom:



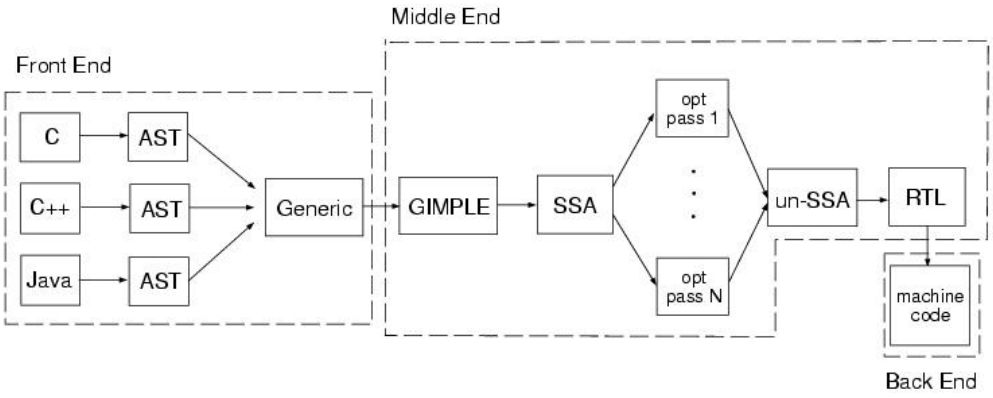
Referências:

https://subscription.packtpub.com/book/game_development/9781783551774/1/ch01lvl1sec10/game-architecture

https://qspace.library.queensu.ca/handle/1974/8552

E) Um compilador é um software que converte uma linguagem fonte (relacionada ao alto nível, ou seja, próxima da linguagem humana – linguagens de programação) para uma linguagem-objeto (relacionada ao baixo nível, ou seja, a um nível que o computador possa executar – binário). Geralmente, esses programas possuem duas fases: a de análise, ou seja, àquela em que o código-fonte será verificado para que não possa haver quaisquer problemas; e a de síntese, em que esse código-fonte será "convertido" para a linguagem de máquina (programa-objeto).

Com relação ao compilador da linguagem C, especialmente o mais famoso e utilizado denominado GCC, tem-se a seguinte arquitetura **linear ("duto e filtro")** de compilação:



Isso se deve ao fato de que este compilador, necessariamente, precisa passar por **cada** etapa individualmente até a geração do programa-objeto, como percebe-se acima: a partir do código-fonte (escrito em C, C++ ou Java) é gerado um “código genérico” associado, tendo este novo código simplificado pelo GIMPLE e convertido para um tipo denominado “árvore SSA”. A partir disso, o SSA é otimizado por mais de 20 tipos de otimização e é gerado o código em RTL, o qual otimiza a árvore para representação baseada em hardware. Por fim, esse RTL é convertido em linguagem de máquina, gerando o programa-objeto.

Referências:

https://cs.lmu.edu/~ray/notes/compilerarchitecture/#:~:text=Compilers%20are%20programs%2C%20and%20generally,analysis%2Dsynthesis%20model%20of%20translation.

https://en.wikibooks.org/wiki/GNU_C_Compiler_Internals/GNU_C_Compiler_Architecture