Secretaria do Planejamento
e das Finanças - SEPLAN

GOVERNO
DO RIO GRANDE DO NORTE

ARQUITETURA DO SISTEMA AMMHC

GRUPO BANCO MUNDIAL

GOVERNO CIDADÃO
DESENVOLVIMENTO E SUSTENTABILIDADE

GOVERNO
DO RIO GRANDE DO NORTE

GRUPO BANCO MUNDIAL

GOVERNO CIDADÃO
DESENVOLVIMENTO E SUSTENTABILIDADE

Este documento é fruto de uma ação estratégica do Governo do Estado do Rio Grande do Norte, através do Projeto Governo Cidadão, financiado com recursos do acordo de empréstimo com o Banco Mundial - BIRD 8276-BR.

L&L SOLUÇÕES

Arquitetura do Sistema AMMHC

Versão 1.0
06/04/2018

L&L Soluções
Rua Desembargador Sinval Moreira Dias, 1749, Lagoa Nova, Natal – RN
Contato: (84) 3207-1069

Histórico de Revisão

Data	Descrição	Autor
06/04/2018	Criação primeira versão do documento	JVA

Sumário

1 - Introdução

1.1 Objetivo

Este documento tem por objetivo descrever a arquitetura do sistema AMMHC mostrando os módulos arquiteturais, fluxo de dados e tecnologias utilizadas. Será explicado também as estratégias adotadas para garantir a segurança dos dados e auditoria do sistema.

1.3 Referências

Documentos de referência

Termo de referência da SDP Nº: 109/2017 – ID 77.CS

Termo de abertura do projeto

Ata de reunião Nº 02/2018 do dia 24/01/2018

Ata de reunião Nº 03/2018 do dia 05/02/2018

Modelo de entidade relacionamento criado pela equipe da EMPARN

2 – Fluxo de dados

Com base nos requisitos e no termo de referência, foi elaborado o seguinte fluxo de dados mostrado na Figura 1.

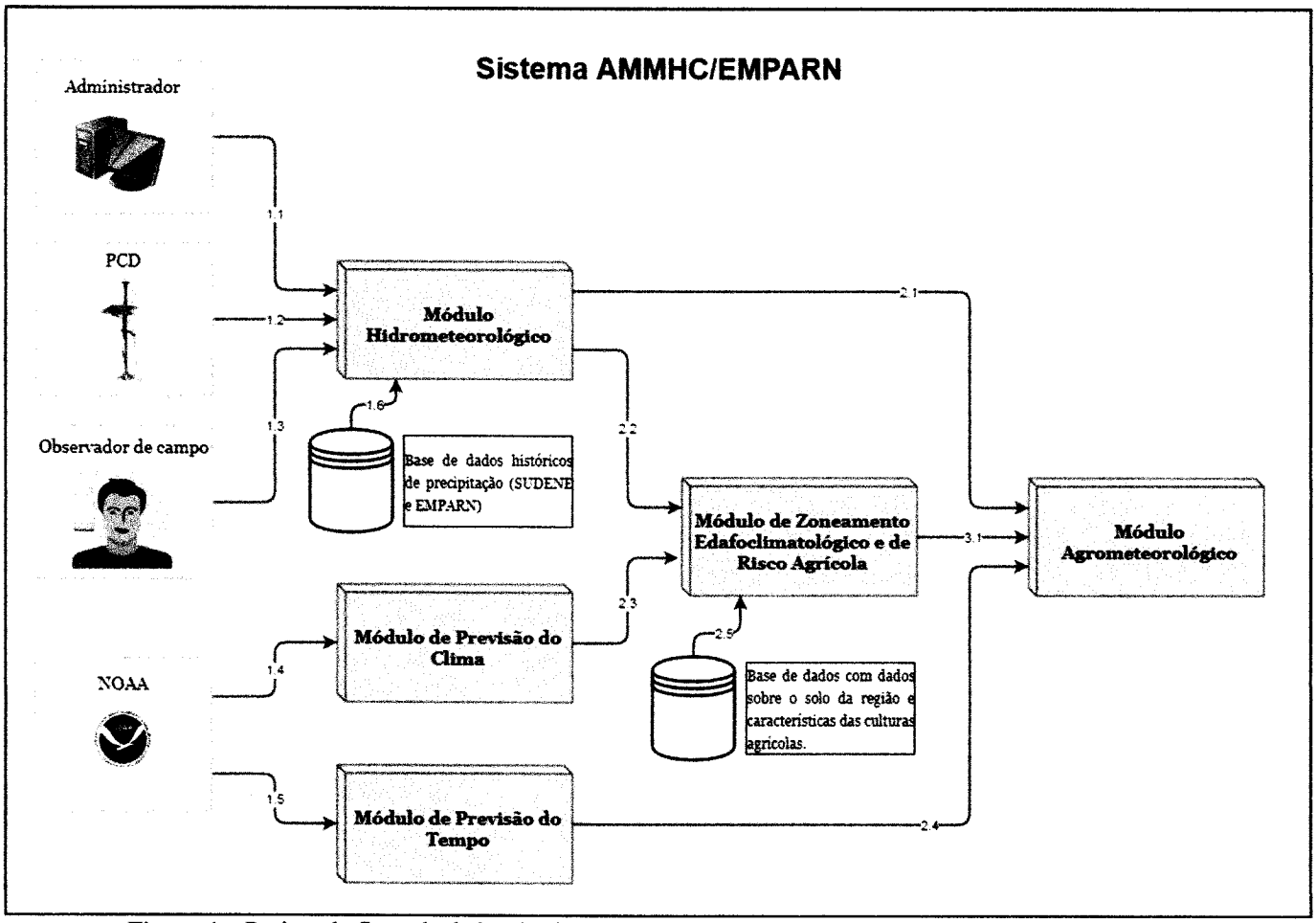

Figura 1 – Projeto de fluxo de dados do sistema AMMHC.

O projeto de fluxo de dados do sistema foi projetado desde a entrada de dados externos até a geração das análises e *insight*. Em consonância com o termo de referência o fluxo de dados atende os cinco módulos pretendidos para o sistema que são: hidrometeorológico; previsão do clima; previsão do tempo; zoneamento edafoclimatológico e risco agrícola; agrometeorológico. A seguir é especificado o fluxo de dados mostrado na Figura 1.

1. **Entradas inicias**

1.1 Chuva (mm).

1.2 Rede de telepluviômetros e Estações Meteorológicas: Chuva (mm), temperatura (°C), vento (m/s), pressão (hPa), umidade do ar (%), umidade do solo (%), radiação solar (kJ/m²) e demais sensores existentes no equipamento de coleta de dados.

1.3 Chuva (mm).

1.4 Arquivo de dados do NOAA.

1.5 Arquivo de dados do NOAA.

1.6 Chuva (mm) - Banco de dados da SUDENE (até 1990) e banco de dados da EMPARN (1993 em diante).

2. **Saída do primeiro nível e entrada do segundo e terceiro nível**

2.1 Chuva (mm), temperatura (°C), vento (m/s), pressão (hPa), umidade do ar (%), umidade do solo (%), radiação solar (kJ/m²), balanço hídrico por município.

2.2 Chuva (mm), temperatura (°C), vento (m/s), pressão (hPa), umidade do ar (%), umidade do solo (%), radiação solar (kJ/m²) por município.

2.3 Previsão de chuva (mm) os valores prováveis de ocorrência de chuvas por município e previsão climática (valor mensal e acumulado para três meses), para todos os municípios descriminando a categoria (abaixo do Normal (mais de -15% abaixo da média histórica do município, Normal, entre -15% a +15% da média histórica do município, e acima do normal acima de 15% da média histórica para o município).

2.4 Temperatura, chuva e vento para um período de até 7 dias por município.

2.5 Dados das características de solo e culturas de cada região.

3. **Saída de segundo nível e entrada do terceiro nível**

3.1 Evapotranspiração real através do balanço hídrico, evapotranspiração potencial decendial e informações sobre capacidade de retenção de água no solo para solos tipo I e II com profundidade de 30 cm e 50 cm.

3 – Arquitetura do Sistema

A fim de atender os requisitos do sistema foi projetada a arquitetura mostradas na Figura 2. Essa arquitetura é dividida em quatro blocos são eles:

- front-end AMMHC WEB: responsável pela visualização dos dados através de acesso via WEB.
- Aplicativos: aplicativos na plataforma Android e iOS para consumir e informar dados em dispositivos móveis.

- back-end AMMHC WEB: responsável por gerenciar os dados de todo o sistema e disponibilizá-los aos demais blocos.
- Execução modelos: responsável por gerenciar e controlar a execução dos modelos de tempo e clima.

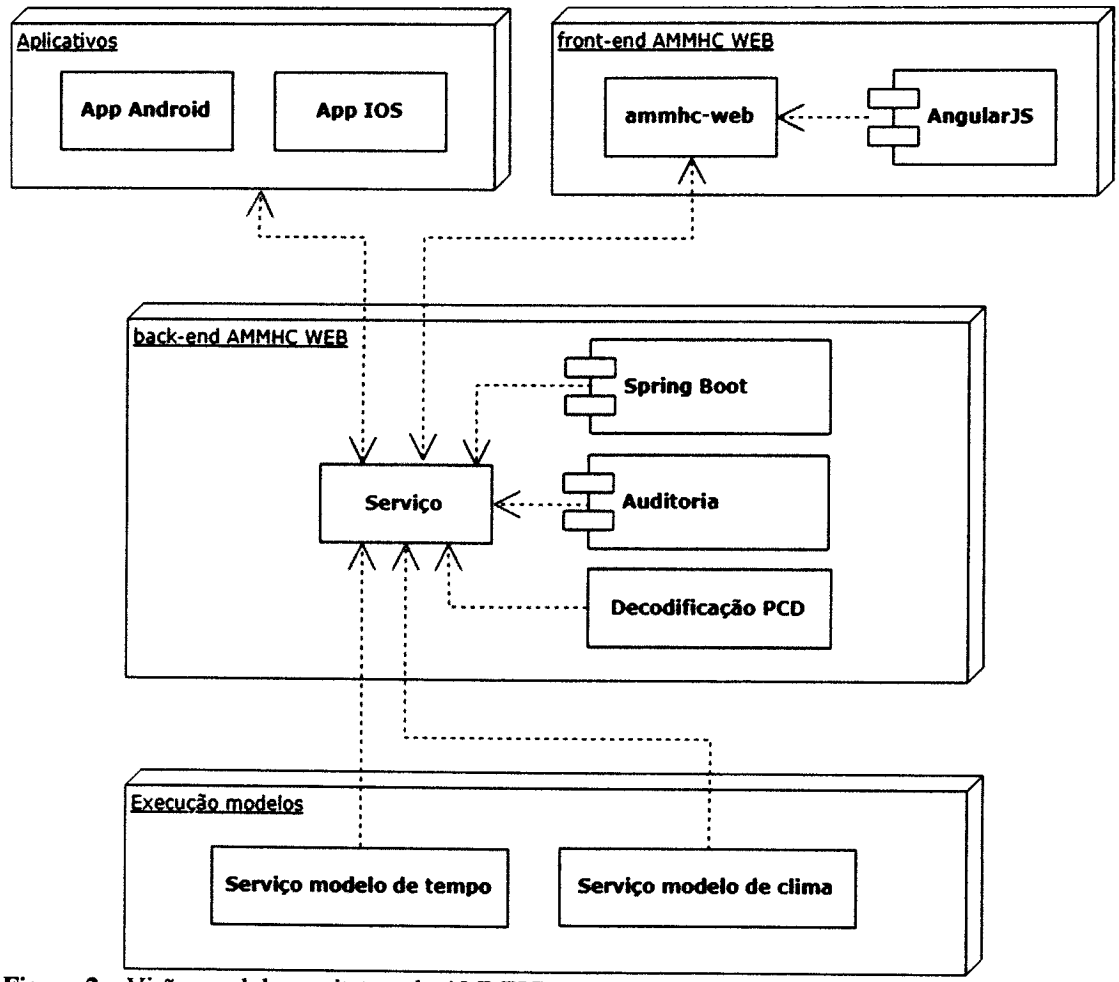

Figura 2 – Visão geral da arquitetura do AMMHC.

3.1 Arquitetura do sistema WEB

A arquitetura do sistema AMMHC foi projetada ser robusta e escalável capaz de suprir a demanda de armazenamento e de acesso aos usuários. Pensando nisso, a arquitetura foi desenvolvida baseada no padrão MCV (*Model View Controller*) onde cada parte do sistema tem sua responsabilidade melhorando assim a organização do sistema e facilitando a manutenibilidade.

O MVC, em português modelo-visão-controlador, é um padrão de arquitetura de software (*design pattern*) que se propõe a dividir um sistema em camadas, cada uma delas com responsabilidades bem definidas.

Por esse padrão, o modelo (*model*) consiste na camada onde será armazenada a representação dos dados e de suas lógicas do negócio. É nesta camada onde também são tratadas questões de armazenamento e validação.

Já a visão (*view*) é a camada responsável pela interação com o usuário através de interfaces onde os dados do modelo são apresentados e são capturadas as entradas dos usuários.

O controlador (*controller*), por sua vez, faz a mediação da entrada convertendo-a em comandos para o modelo ou visão. Ou seja, o controlador é o intermediário entre o modelo e a representação visual permitindo o desacoplamento entre a visão e o modelo.

A Figura 3 ilustra a interação entre as camadas do sistema destacando cada camada e suas interações.

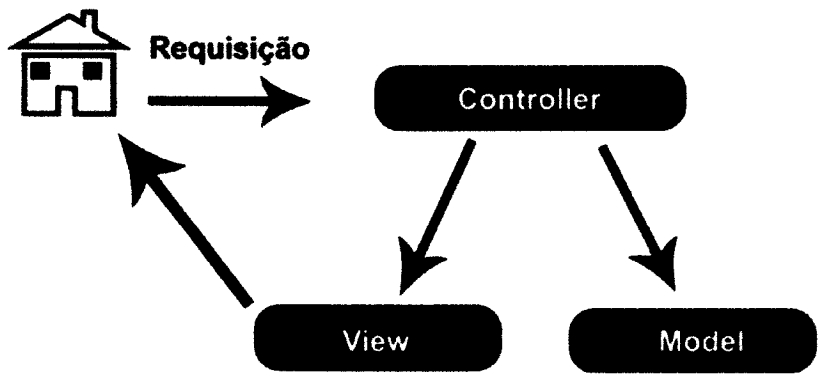

Figura 3 – Diagrama do modelo MVC.

A escolha deste padrão para projeto é indicada, pois o MVC promove a separação de responsabilidades em um projeto de sistemas. Essa separação contribui bastante para a reutilização dos códigos gerados tornando mais simples a expansão e manutenibilidade do sistema. Ainda assim, essa separação em camadas desacopladas

permite que diversas visões sejam criadas sobre os dados, facilitando, por exemplo, ter saídas para dispositivos móveis que será um produto a ser feito em entregas futuras.

Para promover ainda mais uma separação clara entre as camadas criou-se de uma API (*Application Programming Interface*) para que os diversos tipos de aplicações possam interagir de uma maneira padronizada com o sistema e consequentemente com o banco de dados (Figura 4).

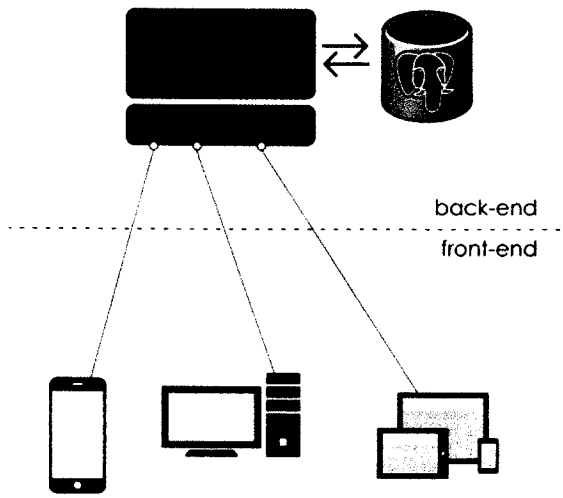

Figura 4 – API para criar separação entre *front-end* e *back-end.*

Com a API, a camada *view* do MVC ficará totalmente desacoplada das outras camadas gerando maior independência entre as camadas garantindo que evoluam separadamente, facilitando a manutenção e evolução da solução.

Para esse tipo de arquitetura, é aconselhado utilizar mecanismos seguros de autenticação através de *tokens* de acesso, que fornece um padrão de autenticação e torna os serviços providos pelas API's independentes dos clientes.

Nesse caso, não são utilizados *cookies* nem sessões do navegador. Em um fluxo normal, tem-se:

1. O usuário informando o ***login*** e uma **senha** na aplicação cliente e enviando para a API

2. A API valida às informações enviadas e caso o usuário seja válido, cria-se um *token* que será retornado para o cliente.
3. Cada requisição feita pelo cliente deve conter no cabeçalho o *token* recebido da API.
4. Caso o *token* obtido no cabeçalho da requisição seja válido, permite-se o acesso à área restrita requisitada.

Com essa estratégia, pode-se ter uma API como *back-end*, provendo serviços a vários clientes (*Web*, Android, IOS, Desktop, etc) e seguindo um padrão único de segurança entre eles.

A tecnologia que será utilizada para implementar esse fluxo de autenticação será o consagrado JSON Web Token (JWT). Diversas tecnologias foram utilizadas para o desenvolvimento do sistema. Assim, nesta seção procuramos destacar as tecnologias mais importantes.

3.1.1 Tecnologias adotadas

Vale salientar a importância de uma análise profunda do *trade-off* entre escolher uma tecnologia muito moderna e tecnologias maduras com baixo potencial de serem abandonadas rapidamente. Trata-se de um problema de todo novo projeto e, neste caso, não seria diferente. Assim, as escolhas aqui apresentadas são escolhas que trarão bastante benefício para o projeto. Modernas em seus conceitos, e ainda assim, maduras e confiáveis para serem utilizadas em um projeto corporativo desta magnitude.

Para introduzir as escolhas posteriormente apresentadas, é importante ter o entendimento que o desenvolvimento de uma aplicação *web* pode ser grosseiramente dividido em dois mundos: a parte que é executada no servidor (*back-end*) e a parte executada no browser do cliente (*front-end*).

A parte do sistema executada no lado do servidor (*back-end*) tem como principal função realizar a interação direta com o modelo de dados e o processamento das regras de negócio. Por ser executada no servidor traz segurança e confidencialidade, pois o usuário final não tem acesso direto aos dados e detalhes das regras da

aplicação. É o *back-end* que é o responsável por interagir diretamente com o banco de dados e na disponibilização dessas informações aos usuários interessados e devidamente autorizados.

Já o *front-end* é a parte do sistema que é executada no browser do cliente e que faz interação direta com o usuário. O *front-end*, em linhas gerais, cuida de aspectos de apresentação e usabilidade do sistema e tem papel importante para a adoção de um sistema, já que é o responsável pela percepção do sistema pelo usuário.

Feita essa breve explanação, apresentamos a seguir as principais tecnologias e linguagens utilizadas para desenvolvimento do sistema em questão.

- Linguagens

Está sendo utilizada a linguagem Java para programação no lado do servidor (*back-end).* O Java é atualmente a linguagem mais utilizada em todo o mundo, em ainda em crescimento nas empresas, através de novas adoções. Uma coisa que se deve mencionar é que hoje o Java não é apenas uma linguagem, mas sim uma plataforma de desenvolvimento grande e robusta.

Para o *front-end* será utilizado um trio de linguagens que são praticamente unanimidades quando se deseja a criação de páginas e sistemas web, a saber: HTML, CSS e Javascript.

O HTML (abreviação para a expressão inglesa *HyperText Markup Language*, que significa Linguagem de Marcação de Hipertexto) é uma linguagem de marcação utilizada na construção de páginas na Web amplamente difundida e utilizada no mundo. Sua primeira versão data do ano de 1991 criada originalmente pelo físico britânico Tim Berners-Lee para comunicação e disseminação das pesquisas entre ele e seu grupo de colegas. Atualmente encontra-se na versão 5.0 que adicionou bastante funcionalidade para melhorar a interatividade e experiência do usuário no uso de aplicações web. O HTML é uma unanimidade para estruturação semântica do conteúdo de uma página web sendo a linguagem básica para a criação de páginas e sistemas acessados via browser.

Já o CSS (*Cascading Style Sheets*) é uma linguagem, criada por Håkon Wium Lie em 1994, utilizada para adicionar estilo (cor, fonte, espaçamento) em um documento escrito em uma linguagem de marcação. Ou seja, para o desenvolvimento de aplicações web, o HTML se preocupa com a organização semântica do documento e o CSS adiciona aspectos visuais para criar páginas web visualmente atraentes. Como vantagem, o CSS permite que os aspectos visuais fiquem desacoplados de seu conteúdo facilitando o reuso e mudanças de cunho estético. Sem o CSS uma simples alteração de cor de fundo de um site/sistema teria que ser replicada manualmente em diversos arquivos dificultando bastante a manutenção principalmente em sistemas maiores compostos por centenas de páginas.

Por fim, o Javascript, frequentemente abreviado como "JS", é uma linguagem de alto nível, dinâmica, não tipada e interpretada em tempo de execução. Ao lado de HTML e CSS, o Javascript é uma das três principais tecnologias de produção de conteúdo da web. As maiorias dos sites o utilizam e todos os navegadores modernos o suportam sem a necessidade de plug-ins. O Javascript é utilizado para dar o aspecto dinâmico as páginas web permitindo ao programador realizar computação de dados ainda na máquina do cliente. Dessa maneira é possível realizar validações antes de submeter os dados ao servidor, realizar requisições assíncronas, atualizar parte do conteúdo de uma página web sem recarregamento, realizar animações, dentre outras facilidades para aumentar a experiência do usuário ao utilizar os sistemas.

- Framework Spring Boot

O Spring Boot é uma solução *open source*, sob a licença MIT, para desenvolvimento de softwares na linguagem Java, esse framework é bastante maduro e muito aplicado na criação de grandes sistemas corporativos, possui um grande ecossistema de ferramentas que ajudam no desenvolvimento do sistema. Seu objetivo não é trazer novas soluções para problemas que já foram resolvidos, mas sim reaproveitar as tecnologias existentes no ecossistema Spring para aumentar a produtividade do desenvolvedor. Trata-se também de uma excelente ferramenta principalmente para adotar-se na escrita de aplicações que fazem uso da arquitetura de micro serviços.

- PostgreSQL

O PostgreSQL é um banco de dados relacional, *open source*, bastante maduro concebido no ano de 1986 na Universidade da Califórnia pelo cientista Michael Stonebraker. Anteriormente era chamado apenas de *Ingress*, no entanto, a partir de 1996 passou a ser chamado de PostgreSQL tendo sua primeira versão oficial em 1996.

Trata-se de um banco de dados extremamente popular, maduro e uma das soluções de código aberta mais robustas para armazenamento relacional de dados. Sem dúvidas é uma escolha coerente para esse projeto, pois não trará custos com licença e ainda assim não deixará em nada a desejar para a plena execução dos requisitos do sistema.

- Angular

Angular é um framework em Javascript, de código aberto e que é mantido pelo Google. Seu objetivo é aumentar aplicativos que podem ser acessados por um navegador web e tem como padrão o MVVM (Model-View-View-Model).

O framework AngularJS funciona através da leitura de páginas HTML, que tem embutido nelas atributos adicionais personalizados em suas tags. Angular interpreta esses atributos como as diretivas para ligar partes de entrada ou saída de página para um modelo que é representado por variáveis em padrão Javascript. Os valores dessas variáveis Javascript podem ser configurados manualmente no código ou recuperado a partir de recursos JSON estáticos ou dinâmicos.

4 – Estratégia de segurança dos dados

O fornecimento de segurança e auditoria de banco de dados sem dúvidas um ponto chave na arquitetura de um sistema desse porte. Sendo assim foi implementado aspectos de segurança e auditoria para garantir a rastreabilidade de todas operações realizadas no sistema.

Nível de Sistema

Como uma camada de segurança dos dados, todas as operações realizadas no banco de dados serão realizadas por intermédio do sistema. Ou seja, o banco de dados está protegido por uma camada de software garantindo que as mudanças realizadas no banco sejam através do sistema. Dessa maneira, o sistema tem o controle total sobre todas as operações realizada no banco de dados e tem a função de registrar essas mudanças.

Como medida de segurança e rastreabilidade dos dados, o sistema registra as revisões no banco de dados. Para isso, existe uma tabela chamada REVINFO com informações das revisões do banco de dados. Essa tabela tem o instante do tempo das alterações e um número sequencial que registra o estado do banco de dados em um determinado instante no tempo.

Para cada tabela do banco de dados foi criado uma tabela adicional para armazenamento de informações de auditoria informando os dados da revisão e o tipo de alteração realizado (Adição, Alteração ou Remoção). Na tabela da revisão há replicação dos dados como forma de garantir que mesmo em uma remoção os dados possam ser rastreados e uma operação de rollback seja possível. A Figura 5 mostra um exemplo onde duas tabelas de dados são criadas para o modelo (Cliente e Coche) e as respectivas tabelas adicionais para registro de atividade.

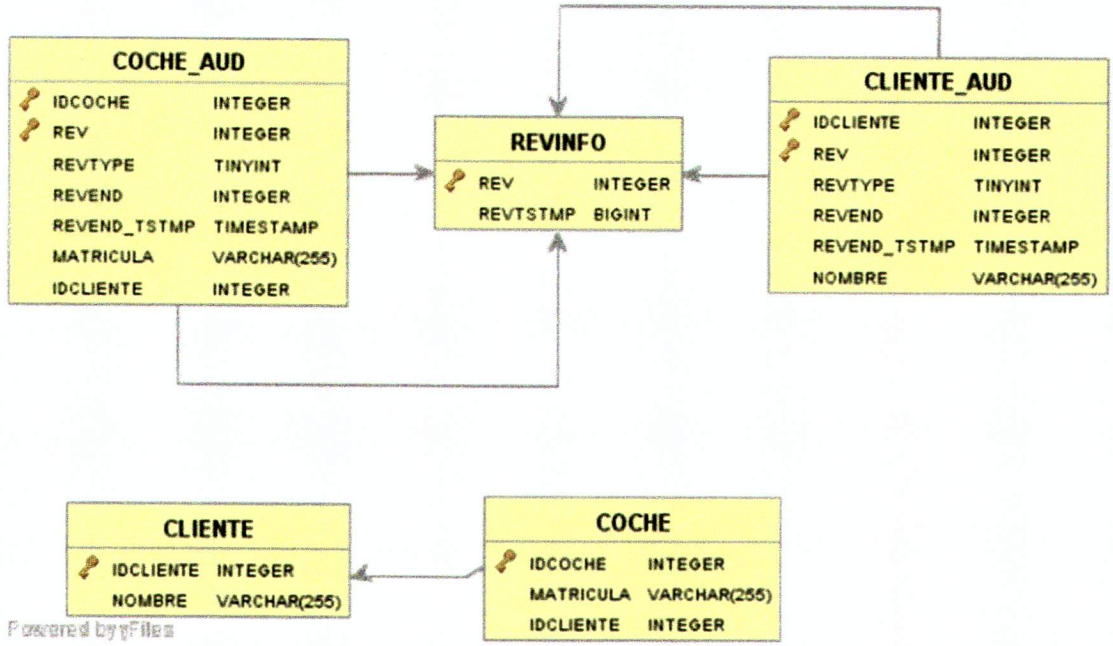

Figura 5 – Exemplo modelagem de dados com registro de mudança
Fonte: *https://jrodriguezweb.wordpress.com/2013/10/29/hibernate-envers-ejemplo/*

As Tabelas abaixo mostram exemplo dos dados da tabela CLIENTE e CLIENTE_AUD simulando registro de entradas auditáveis.

Tabela 1 Exemplo Tabela CLIENTE

IDCLIENTE	NOMBRE
123	John
124	Pitt

Tabela 2 Exemplo de Tabela CLIENT_AUD

REV	IDCLIENTE	NOMBRE	REVTYPE
64	123	John	ADD
65	124	Pitt	MOD
66	125	Smith	DEL

Assim, com a implementação dessa estratégia é possível ter total rastreabilidade do banco de dados garantindo maior segurança aos dados e facilitando análise de intervenções indevidas ou por bugs do sistema.

Nível de Banco de dados

Para um grau de auditoria e segurança ainda mais detalhado, os logs do banco de dados PostgreSQL foi configurado para que as operações sejam salvas nos arquivos de Log, esses arquivos são carregados para o Elasticsearch através da ferramenta Logstash, após esses dados estarem no Elasticsearch eles podem ser consultados e analisados utilizando a ferramenta Kibana. Assim, teremos uma segurança adicional para caso de intervenção direta no banco de dados por algum software/sistema inicialmente não autorizado.

4 – Modelo WRF

O modelo numérico escolhido para previsão do tempo foi o WRF (*Weather Research Forecasting*), ele possui código aberto e passível de download, que atualmente é bastante usado em pesquisas e previsão do tempo sobre a região Nordeste do Brasil (NEB) trazendo resultados satisfatórios e consistentes perante a comunidade, além de seu fórum de discussão atualizado periodicamente.

O WRF é um sistema de modelagem dinâmica da atmosfera usada em centros operacionais de Meteorologia, Agricultura e na Aeronáutica. Foi desenvolvido por colaboradores de universidades, centros operacionais e agências governamentais, especialmente nos Estados Unidos: a *Mesoscale and Microscale Meteorology* (MMM) *Division National Center for Atmosfpheric Research* (NCAR), *National Centers for Environmental Prediction* (NCEP) e o *Forecast Sytem Laboratory* (FSL), pertencentes à *National Oceanic and Atmospheric Administration* (NOAA), a *Air Force Weather Agency* (AFWA) e o *Naval Research Laboratory* (NRL), tutelados pelos *US Departement of Defense*, o *Center of Analysis and Prediction of Storms* (CAPS), sediado na Universidade de Oklahoma, e a *Federal Aviation Administration* (FAA).

Devido ao número elevado de desenvolvedores dessas instituições é possível atualizar o código fonte do modelo a cada quatro meses, fazendo dele o estado da arte em simulação de processos atmosféricos. Além disso, o modelo é flexível, portável e eficiente em várias plataformas da computação.

O WRF foi implementado no *cluster* computacional da EMPARN. O código do modelo é escrito em linguagem computacional *Fortran90* e pode ser obtido através do portal www.mmm.ucar.edu/wrf/users. Ele foi desenvolvido com a possibilidade de ser acoplado a outros modelos, como o de dispersão de poluentes, hidrológicos e de escoamento de microescala. Shamarock et al., (2008) descreveram com detalhes a versão 3 do modelo.

O núcleo (*core*) de qualquer modelo numérico de simulação hidrodinâmico consiste na formulação das equações dinâmicas apropriadas, juntamente com as técnicas empregadas na resolução das mesmas. O sistema de modelagem WRF contempla dois núcleos dinâmicos distintos: *Advanced Research* WRF (ARW) e o *Nonhydrostatic Mesoscale Model* (NMM).

O NMM é um modelo não hidrostático desenvolvido pelo NCEP a partir do modelo operacional hidrostático ETA. O ARW tem um esquema de divisão do incremento de integração para as ondas acústicas e de gravidade oriundo do modelo de nuvens de Klemp-Whilhelmson. Possuem diferenças significativas quanto à formulação das equações dinâmicas, as variáveis prognósticas usadas, e ao seu modo como são dispostas na malha (*grid staggering*), e quanto aos métodos de integração temporal. Shamarock et al., (2008) explicaram as diferenças e as semelhanças, apontando para a viabilidade de uma unificação futura dos dois núcleos. O WRF-ARW é de responsabilidade do NCAR. O WRF-NMM está em cargo do NCEP/NOAA.

O núcleo escolhido e instalado na EMPARN para previsão do tempo é o ARW, versão 3.9.1.1. Para uma descrição mais detalhada dos seus aspectos sugere-se uma consulta ao documento ARW *User's Guide* e o NCAR *Technical Note: A Description of the Advanced Research* WRF Version 3.9, obtido no mesmo portal do modelo. O segundo documento explica os fundamentos físicos e matemáticos da formulação dinâmica do modelo, parametrizações físicas e as respectivas referências dos trabalhos originais dos seus autores. O WRF-ARW é um modelo não hidrostático, com opção hidrostática, de previsão numérica de tempo de mesoescala. Apresenta quatro módulos principias, o WPS (WRF *Preprocessing System*), o WRF-Var, o ARW (*Advanced Research* WRF) e o ARWPost.

No módulo WPS, que é compilado e usado para a preparação dos dados necessários às simulações (variáveis estáticas da geografia do terreno e uso do solo e as variáveis meteorológicas). Além disso, definem-se os domínios de cálculo, interpolam os dados de superfície (topografia e uso do tipo de solo) e interpolam os dados meteorológicos de fontes externas. A composição física da superfície terrestre do modelo será configurada por meio dos dados da topografia USGS (*United States Geological Survey*) (Figura 6) e vegetação MODIS (*Moderate Resolution Imaging Spectroradiometer*) (Figura 7), ambos com resolução de 30 arc-segundos, ou 925 m.

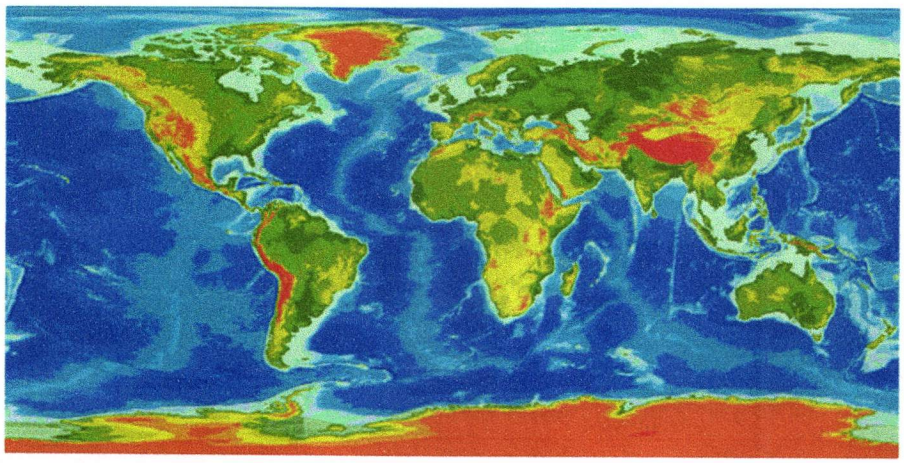

Figura 6 – Dado topográfico USGS-30arc-segundo global utilizado no modelo WRF da simulação para sua área de domínio.
Fonte:https://woodshole.er.usgs.gov/openfile/of2005-1001/data/basemaps/srtm30plus/srtm30plus-world_pctshade.gif

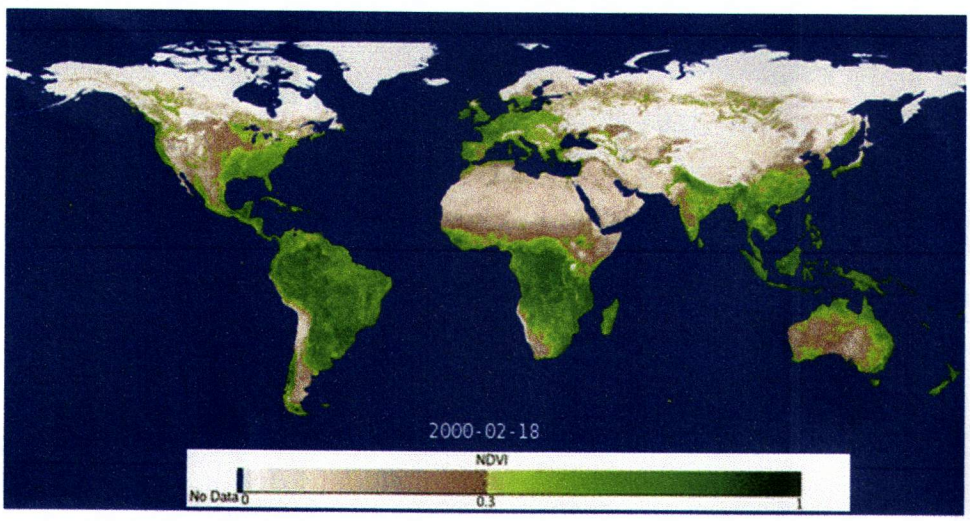

Figura 7 – Dado da vegetação MODIS-30arc-segundo global utilizado no modelo WRF da simulação para sua área de domínio.
Fonte:
https://lpdaac.usgs.gov/user_resources/data_in_action/modis_global_vegetation_index_animation.

Os dados geográficos com sua caracterização de uso do solo, vegetação e topográficos são obtidos no mesmo portal do modelo WRF. Logo, encontram-se todos os arquivos necessários para simulação do modelo com diferentes resoluções espacial e temporal por meio dos dados estáticos geográficos. O WPS é formado por três programas executáveis. Após toda compilação realizada neste modulo serão obtidos três arquivos executáveis: (i) o *geogrid.exe*, que faz a preparação dos dados orográficos; (ii) o *ungrid.exe*, que decodifica os dados globais e (iii) o *metgrid.exe*, que interpolam os níveis verticais do modelo. O módulo WRF-Var é opcional, sendo utilizado apenas para inserir dados observacionais das análises interpoladas pelo WPS, mas não foi implementado. O módulo ARW é a parte principal do modelo WRF. Nele contém as opções da física do modelo e é composto por vários subprogramas após sua compilação (*ideal.exe*, *real.exe* e *wrf.exe*), que realizam a inicialização e a posterior simulação.

A discretização espacial utilizada pelo núcleo de processamento ARW é do tipo C (Arakawa e Lamb, 1977), conhecido como "arranjo deslocado". As variáveis dinâmicas, que são as componentes escalares da velocidade do escoamento (U, V, W) são calculadas no ponto médio do elemento de grade, enquanto as outras propriedades termodinâmicas (chamados na **Figura** 8 de θ) são calculadas no centro da malha. Neste esquema, a discretização espacial é resolvida com a utilização do método matemático de diferenças finitas com *upwind* de 5ª ordem ou diferenças finitas de 6ª ordem.

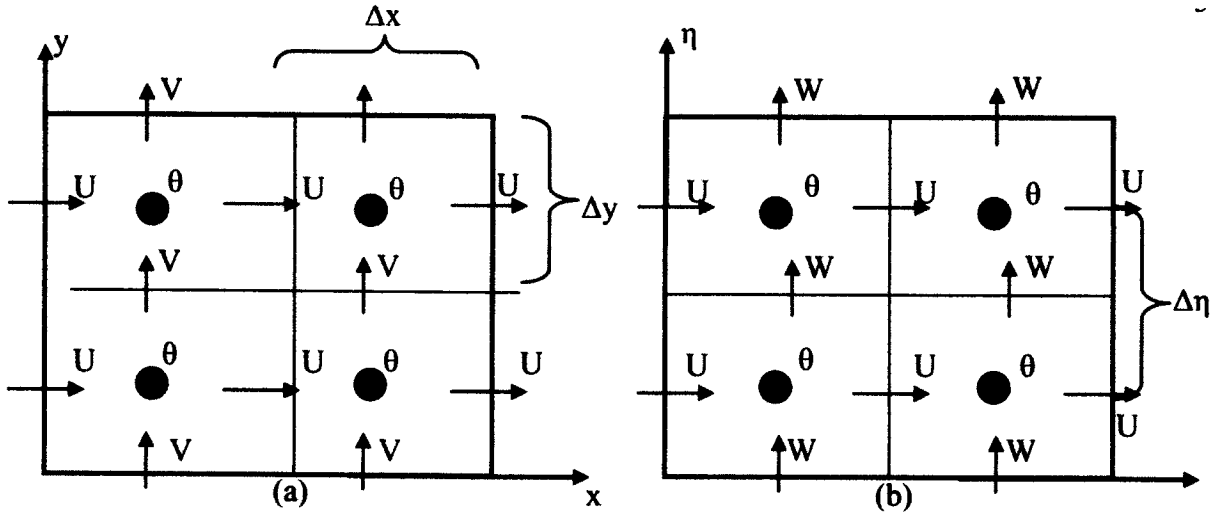

Figura 8 - Malha horizontal (a) e vertical (b) do WRF-ARW.
Fonte: Adaptado de Skamarock et al., (2008).