Simulador Elevadores

João Lucas Veloso & Eike Daniel

Na contemporaneidade, observa-se uma crescente demanda por soluções que aliem comodidade à sustentabilidade. A busca por práticas sustentáveis vai além da preservação ambiental, abrangendo também a otimização no uso eficiente de recursos naturais, como a energia elétrica. No Brasil, as edificações — englobando os setores residencial, comercial e público — são responsáveis por aproximadamente 52% do consumo total de eletricidade do país (MEGAWHAT, 2024), superando o setor industrial. Dentro desse cenário, os edifícios comerciais e públicos utilizam majoritariamente a eletricidade, com 92% de participação em sua matriz energética, enquanto as residências utilizam 46% de eletricidade, 26% de GLP e 24% de lenha (EIXOS INTELIGÊNCIA, 2024).

Embora não haja um número exato disponível sobre a quantidade de elevadores em operação no Brasil, é evidente que esses equipamentos desempenham um papel significativo no consumo energético de edificações, especialmente em prédios comerciais e residenciais de grande porte. Considerando a necessidade de racionamento de energia elétrica e a importância de otimizar mecanismos que proporcionem conforto de forma sustentável, torna-se essencial o desenvolvimento de sistemas capazes de reduzir o consumo de energia elétrica e atender eficientemente aos usuários por meio da gestão inteligente dos serviços de elevação predial.

A mobilidade interna em edificações de múltiplos andares configura-se como um dos principais desafios no âmbito da engenharia de controle e automação predial. Nesse contexto, os sistemas de transporte vertical, especialmente os elevadores, desempenham um papel fundamental na garantia da eficiência operacional e na melhoria da experiência dos usuários. Mais do que promover a simples locomoção entre pavimentos, esses sistemas devem ser projetados para atender de forma ágil, segura e eficiente às diversas demandas, contemplando não apenas o fluxo geral de passageiros, mas também assegurando acessibilidade e priorização no atendimento a pessoas com mobilidade reduzida, como idosos e cadeirantes. Dessa forma, torna-se indispensável que as soluções adotadas conciliem desempenho, conforto, segurança e inclusão, alinhando-se às exigências atuais de sustentabilidade e qualidade dos ambientes construídos.

1. **Main.java**

Descrição: Classe principal que inicializa e executa toda a simulação.

Principais Funcionalidades da Classe Main:

1. Inicialização do Sistema
   - Cria a configuração inicial do sistema
   - Inicializa o controlador de elevadores
   - Configura o painel visual ASCII
   - Prepara o ambiente para a simulação

2. Controle Temporal

- Define a duração total da simulação (1440 minutos = 24 horas)
- Gerencia o ciclo de simulação minuto a minuto
- Identifica horários de pico (7-9h e 17-19h)
- Controla o fluxo de tempo da simulação

3. Gerenciamento de Log
- Cria e mantém o arquivo de log da simulação
- Registra informações detalhadas de cada ciclo
- Armazena dados sobre o estado do sistema
- Mantém histórico completo da simulação

4. Coordenação de Ciclos
Para cada minuto da simulação:
- Gera cabeçalho com horário e status
- Atualiza o estado dos elevadores
- Registra consumo de energia
- Atualiza o painel visual
- Salva informações no log

5. Monitoramento de Progresso
- Exibe porcentagem de conclusão da simulação
- Mostra progresso a cada 100 ciclos
- Permite acompanhamento do andamento
- Facilita o monitoramento da simulação

6. Apresentação de Dados
- Mostra status dos elevadores
- Exibe consumo de energia por ciclo
- Apresenta total de energia gasta
- Atualiza o painel visual em tempo real

7. Geração de Resumo

- Produz resumo final da simulação
- Apresenta estatísticas gerais
- Mostra resultados consolidados
- Indica localização do arquivo de log

8. Controle de Fluxo
- Coordena a execução dos diferentes componentes
- Garante sincronização entre elementos
- Mantém a ordem correta das operações
- Controla o fluxo da simulação

9. Interface com Usuário
- Solicita confirmação para iniciar
- Mostra mensagens de progresso
- Exibe informações em tempo real
- Apresenta resultados finais

10. Tratamento de Erros
- Gerencia exceções durante a simulação
- Trata erros de escrita no log
- Mantém a simulação funcionando mesmo com problemas
- Reporta erros quando necessário

11. Verificação de Horários
- Identifica

2. **Configuracao.java**

Descrição: Classe que concentra os parâmetros configuráveis do sistema.

Principais Funcionalidades da Classe Configuracao

1. Definição de Parâmetros do Sistema
   - Número de andares do prédio
   - Quantidade de elevadores disponíveis
   - Capacidade máxima de passageiros por elevador
   - Peso máximo suportado por elevador

2. Configuração de Tempos
   - Tempo mínimo de viagem entre andares (em segundos)
   - Tempo máximo de viagem entre andares (em segundos)
   - Tempo máximo de espera permitido (em minutos)

3. Parâmetros de Energia
   - Consumo de energia por deslocamento entre andares
   - Consumo de energia por parada do elevador
   - Unidades de energia utilizadas no sistema

4. Valores Padrão
   - Inicializa o sistema com configurações padrão:
     - 10 andares
     - 2 elevadores
     - Capacidade de 10 passageiros
     - Tempo mínimo de viagem: 5 segundos
     - Tempo máximo de viagem: 10 segundos
     - Tempo máximo de espera: 30 minutos
     - Consumo de energia por andar: 2 unidades

- Consumo de energia por parada: 1 unidade
- Peso máximo: 1000 kg

5. Métodos de Acesso
- getCapacidadeMaximaPassageiros(): Retorna a capacidade máxima de passageiros
- getCapacidadeMaximaPeso(): Retorna o peso máximo suportado
- getPesoMaximo(): Retorna o peso máximo do elevador
- getConsumoPorAndar(): Retorna o consumo de energia por andar
- getAndarMaximo(): Retorna o número do último andar

6. Gerenciamento de Limites
- Define limites operacionais do sistema
- Estabelece restrições de capacidade
- Controla parâmetros de performance
- Define limites de segurança

7. Configuração de Performance
- Estabelece tempos de operação
- Define limites de espera
- Configura parâmetros de eficiência
- Controla consumo de energia

Esta classe é fundamental para o sistema, pois:
- Define todos os parâmetros operacionais
- Estabelece limites do sistema
- Fornece configurações padrão
- Permite ajustes nos parâmetros
- Controla a performance do sistema
- Define restrições de segurança
- Estabelece parâmetros de eficiência

A classe Configuracao serve como base para toda a simulação, definindo como o sistema deve operar e quais são seus limites operacionais.

3. **ControladorElevadores.java**

Descrição: Responsável pelo gerenciamento global do sistema e da simulação.

Principais Funcionalidades da Classe ControladorElevadores

1. Gerenciamento do Sistema
   - Controla todos os elevadores do sistema
   - Gerencia as filas de espera em cada andar
   - Coordena o painel de controle
   - Mantém o resumo da simulação

2. Inicialização do Sistema
   - Cria os elevadores com base na configuração
   - Inicializa as filas de prioridade para cada andar
   - Configura o painel de controle
   - Prepara o resumo da simulação

3. Simulação de Ciclos
   - Incrementa o tempo de espera das pessoas nas filas
   - Movimenta os elevadores
   - Calcula o consumo de energia por ciclo
   - Atualiza o painel de controle

4. Geração de Pessoas
   - Gera pessoas em horários normais e de pico
   - Define características aleatórias para cada pessoa:
     - Idade (10 a 80 anos)
     - Status (cadeirante, idoso, normal)
     - Peso (40 a 100 kg)
   - Distribui pessoas entre andares
   - Registra pessoas geradas no resumo

5. Gerenciamento de Filas
   - Mantém filas separadas para subida e descida
   - Atualiza o tempo de espera das pessoas
   - Gerencia a inserção de novas pessoas
   - Controla a prioridade de atendimento

6. Controle de Elevadores
   - Coordena a movimentação dos elevadores
   - Gerencia o atendimento de chamadas
   - Controla o consumo de energia
   - Monitora o estado de cada elevador

7. Painel de Controle
   - Atualiza os botões de chamada
   - Ativa/desativa botões conforme necessidade
   - Mantém o estado dos botões atualizado
   - Coordena as chamadas dos elevadores

8. Registro de Estatísticas
   - Mantém contagem de passageiros
   - Registra consumo de energia
   - Controla tempo de espera
   - Gera resumo da simulação

9. Horários de Pico
   - Identifica horários de pico (7-9h e 17-19h)
   - Aumenta a geração de pessoas nesses períodos
   - Adapta o sistema para maior demanda
   - Otimiza o atendimento

10. Métodos de Acesso
    - Fornece acesso aos elevadores
    - Permite consulta às filas
    - Disponibiliza o painel de controle
    - Acessa o resumo da simulação

11. Resumo Final
    - Gera relatório completo da simulação
    - Apresenta estatísticas gerais
    - Mostra resultados consolidados
    - Fornece insights sobre o sistema

Esta classe é o coração do sistema, pois:
- Coordena todas as operações
- Gerencia os recursos do sistema
- Controla o fluxo de pessoas
- Mantém o sistema funcionando
- Coleta dados para análise
- Otimiza o funcionamento
- Garante a eficiência do sistema

O ControladorElevadores é responsável por orquestrar todo o funcionamento do sistema de elevadores, garantindo que todas as partes trabalhem em harmonia e que o sistema opere de forma eficiente e segura.

4. **Elevador.java**

Descrição: Representa um elevador individual no sistema.

Principais Funcionalidades da Classe Elevador

1. Gerenciamento de Estado
   - Controla o andar atual do elevador
   - Gerencia a direção de movimento
     (-1: descendo, 0: parado, 1: subindo)
   - Mantém registro dos passageiros
   - Controla peso e capacidade atual

2. Controle de Passageiros
   - Gerencia o embarque de passageiros
   - Controla o desembarque no andar de destino
   - Mantém registro do número de passageiros
   - Verifica limites de capacidade e peso

3. Movimentação
   - Controla a movimentação entre andares
   - Calcula tempo de viagem entre andares
   - Atualiza a direção do elevador
   - Gerencia paradas e partidas

4. Atendimento de Chamadas
   - Procura próxima chamada quando vazio
   - Prioriza chamadas de pessoas com necessidades especiais
   - Calcula distância até chamadas
   - Otimiza rotas de atendimento

5. Gerenciamento de Energia
   - Calcula consumo de energia por ciclo
   - Registra energia gasta em deslocamentos
   - Controla energia gasta em paradas
   - Mantém estatísticas de consumo

6. Priorização de Atendimento
   - Identifica pessoas com prioridade (cadeirantes, idosos)
   - Atende chamadas prioritárias primeiro
   - Mantém ordem de atendimento
   - Gerencia filas de espera

7. Verificações de Segurança
   - Controla limite de peso
   - Verifica capacidade máxima
   - Garante segurança dos passageiros
   - Previne sobrecarga

8. Estatísticas e Registros
   - Mantém contagem de embarques/desembarques
   - Registra tempo de viagem
   - Controla energia consumida
   - Gera relatórios de operação

9. Métodos de Acesso
   - Fornece informações sobre estado atual
   - Permite consulta de capacidade
   - Disponibiliza dados de passageiros
   - Acessa estatísticas de operação

10. Otimização de Rotas
    - Calcula melhor rota para atendimento
    - Minimiza tempo de espera

- Otimiza consumo de energia
- Melhora eficiência operacional

11. Controle de Direção
- Atualiza direção baseado em destino
- Gerencia mudanças de direção
- Otimiza movimentação
- Evita movimentos desnecessários

12. Interação com Sistema
- Comunica com painel de controle
- Interage com filas de espera
- Atualiza resumo da simulação
- Coordena com outros elevadores

Esta classe é fundamental para o sistema pois:
- Controla a operação do elevador
- Gerencia passageiros
- Otimiza movimentação
- Controla consumo de energia
- Garante segurança
- Mantém estatísticas
- Prioriza atendimento
- Coordena com o sistema

O Elevador é o componente principal do sistema, responsável por toda a operação de transporte de pessoas entre andares, garantindo eficiência, segurança e conforto para os usuários.

5. **Pessoa.java**

Descrição: Representa um passageiro que utiliza o sistema de elevadores.

Principais Funcionalidades da Classe Pessoa

1. Atributos Básicos
   - Nome da pessoa
   - Idade
   - Status de cadeirante
   - Peso
   - Andar de origem
   - Andar de destino
   - Tempo de espera

2. Gerenciamento de Prioridades
   - Identifica pessoas com prioridade (idosos e cadeirantes)
   - Define níveis de prioridade
   - Fornece texto descritivo da prioridade
   - Permite verificação rápida de status

3. Controle de Tempo
   - Registra tempo de espera
   - Permite incremento do tempo
   - Mantém histórico de espera
   - Facilita análise de eficiência

4. Verificações de Status
   - Verifica se é idoso (60 anos ou mais)
   - Verifica se é cadeirante
   - Determina se tem prioridade

- Fornece informações de status

5. Métodos de Acesso
   - Fornece acesso aos dados pessoais
   - Permite consulta de informações
   - Disponibiliza dados de origem/destino
   - Acessa tempo de espera

6. Representação Textual
   - Gera descrição completa da pessoa
   - Inclui todos os atributos relevantes
   - Mostra status de prioridade
   - Exibe tempo de espera

7. Gerenciamento de Dados
   - Armazena informações pessoais
   - Mantém dados de viagem
   - Controla status de prioridade
   - Gerencia tempo de espera

8. Identificação de Prioridades
   - Define prioridade para idosos
   - Define prioridade para cadeirantes
   - Combina diferentes tipos de prioridade
   - Facilita atendimento preferencial

9. Controle de Viagem
   - Mantém registro de origem
   - Controla destino
   - Gerencia trajeto
   - Facilita planejamento de rota

10. Estatísticas e Análise

- Fornece dados para análise
- Permite cálculo de tempos
- Facilita geração de relatórios
- Ajuda na otimização do sistema

Esta classe é fundamental para o sistema pois:
- Representa os usuários do sistema
- Define prioridades de atendimento
- Controla tempo de espera
- Fornece dados para análise
- Facilita gestão de filas
- Permite otimização do sistema
- Ajuda na tomada de decisões
- Melhora a eficiência do atendimento

A classe Pessoa é essencial para o funcionamento do sistema, pois:
- Representa os usuários que utilizam os elevadores
- Define como cada pessoa deve ser atendida
- Permite priorização de atendimento
- Fornece dados para análise de eficiência
- Ajuda na otimização do sistema
- Facilita a gestão de filas
- Melhora a experiência do usuário
- Contribui para a eficiência operacional

A classe Pessoa é um componente central do sistema, pois todas as decisões de atendimento e priorização são baseadas nas características e necessidades das pessoas que utilizam os elevadores.

6. **FilaPrioridadeDupla.java**

Descrição: Gerencia as filas de espera em cada andar, separando pedidos para subir e descer.

Principais Funcionalidades da Classe FilaPrioridadeDupla

1. Gerenciamento de Filas Duplas
   - Mantém duas filas separadas: uma para subir e outra para descer
   - Gerencia filas com prioridade
   - Controla capacidade de cada fila
   - Coordena operações entre as filas

2. Inserção de Pessoas
   - Insere pessoas na fila apropriada baseado no destino
   - Verifica direção do movimento (subir/descer)
   - Mantém ordem de prioridade
   - Gerencia capacidade das filas

3. Remoção de Pessoas
   - Remove pessoas da fila de subida
   - Remove pessoas da fila de descida
   - Mantém ordem de prioridade
   - Gerencia saída de pessoas

4. Verificação de Estado
   - Verifica se fila de subida está vazia
   - Verifica se fila de descida está vazia
   - Verifica se há pessoas esperando
   - Controla estado das filas

5. Controle de Tempo
   - Incrementa tempo de espera em ambas as filas
   - Mantém registro de espera
   - Atualiza estatísticas
   - Controla eficiência do sistema

6. Contagem de Pessoas
   - Conta pessoas esperando para subir
   - Conta pessoas esperando para descer
   - Conta pessoas por andar
   - Fornece estatísticas de uso

7. Acesso aos Dados
   - Fornece array de pessoas da fila de subida
   - Fornece array de pessoas da fila de descida
   - Permite consulta de estado
   - Facilita análise de dados

8. Impressão de Status
   - Mostra estado da fila de subida
   - Mostra estado da fila de descida
   - Gera logs de operação
   - Facilita debug

9. Gerenciamento de Prioridades
   - Mantém ordem de prioridade em cada fila
   - Respeita necessidades especiais
   - Otimiza atendimento
   - Garante eficiência

10. Análise de Dados
    - Permite contagem por andar
    - Facilita geração de estatísticas

- Ajuda na otimização
- Fornece insights operacionais

Esta classe é fundamental para o sistema pois:
- Gerencia filas de espera
- Controla prioridades
- Otimiza atendimento
- Mantém ordem
- Facilita operação
- Melhora eficiência
- Fornece dados
- Ajuda na tomada de decisões

A classe FilaPrioridadeDupla é essencial para:
- Organizar pessoas esperando
- Priorizar atendimento
- Controlar fluxo de pessoas
- Otimizar operação
- Melhorar eficiência
- Facilitar gestão
- Fornecer dados
- Ajudar na análise

7. **FilaPrioridade.java**

Descrição: Implementa a lógica de fila com prioridade para passageiros.

Principais Funcionalidades da Classe FilaPrioridade

1. Gerenciamento de Fila
   - Mantém array de pessoas
   - Controla quantidade de pessoas
   - Gerencia capacidade da fila
   - Organiza pessoas por prioridade

2. Inserção Ordenada
   - Insere pessoas mantendo ordem de prioridade
   - Verifica capacidade da fila
   - Ordena por nível de prioridade
   - Mantém fila organizada

3. Remoção de Pessoas
   - Remove pessoa com maior prioridade
   - Mantém ordem da fila
   - Atualiza quantidade
   - Gerencia saída de pessoas

4. Controle de Prioridades
   - Define níveis de prioridade:
     - Nível 3: Cadeirantes
     - Nível 2: Idosos
     - Nível 1: Pessoas normais
     - Compara prioridades
     - Mantém ordem correta
     - Garante atendimento preferencial

5. Verificação de Estado
   - Verifica se fila está vazia

- Controla quantidade de pessoas
- Monitora capacidade
- Gerencia estado da fila

6. Controle de Tempo
- Incrementa tempo de espera
- Atualiza para todas as pessoas
- Mantém registro de espera
- Controla eficiência

7. Acesso aos Dados
- Fornece array de pessoas
- Permite consulta de quantidade
- Disponibiliza dados da fila
- Facilita análise

8. Impressão de Status
- Mostra estado da fila
- Lista todas as pessoas
- Gera logs de operação
- Facilita debug

9. Gerenciamento de Capacidade
- Define capacidade inicial
- Verifica limite da fila
- Controla overflow
- Garante funcionamento

10. Operações de Array
- Cria cópias do array
- Mantém ordem dos elementos
- Gerencia espaço
- Otimiza operações

8. **ResumoSimulacao.java**

Descrição: Coleta e armazena as estatísticas da simulação.

Principais Funcionalidades da Classe ResumoSimulacao

1. Contagem de Passageiros
   - Registra total de passageiros gerados
   - Controla número de embarques
   - Contabiliza desembarques
   - Mantém estatísticas de uso

2. Controle de Viagens
   - Registra total de viagens realizadas
   - Monitora movimentação dos elevadores
   - Contabiliza deslocamentos
   - Acompanha operação

3. Gerenciamento de Energia
   - Registra energia gasta total
   - Acumula consumo por ciclo
   - Controla eficiência energética
   - Monitora gastos

4. Controle de Tempo
   - Registra tempo de espera total
   - Calcula tempo médio de espera
   - Monitora eficiência
   - Avalia performance

5. Registro de Eventos
   - Registra geração de passageiros
   - Contabiliza embarques
   - Controla desembarques
   - Monitora viagens

6. Cálculos Estatísticos
   - Calcula tempo médio de espera
   - Totaliza energia consumida
   - Computa estatísticas gerais
   - Fornece métricas

7. Métodos de Registro
   - registrarPassageiroGerado()
   - registrarEmbarque()
   - registrarDesembarque()
   - registrarViagem()
   - registrarEnergiaGasta()

8. Métodos de Consulta
   - getEnergiaGastaTotal()
   - getTempoEsperaMedio()
   - Fornece estatísticas
   - Disponibiliza dados

9. Geração de Relatórios
   - Imprime resumo completo
   - Mostra estatísticas gerais
   - Apresenta métricas
   - Facilita análise

10. Inicialização

- Inicia contadores em zero
- Prepara estrutura de dados
- Inicializa variáveis
- Prepara para coleta

Esta classe é fundamental para o sistema pois:
- Mantém estatísticas
- Controla métricas
- Fornece dados
- Facilita análise
- Avalia performance
- Monitora eficiência
- Gera relatórios
- Ajuda na tomada de decisões

A classe ResumoSimulacao é essencial para:
- Avaliar desempenho
- Analisar eficiência
- Identificar gargalos
- Otimizar operação
- Melhorar sistema
- Tomar decisões
- Gerar relatórios
- Monitorar resultados

A classe ResumoSimulacao é um componente central do sistema, pois:
- Fornece visão geral
- Permite análise
- Facilita otimização
- Ajuda na gestão
- Melhora eficiência
- Monitora resultados

- Gera insights
- Suporta decisões

Esta classe é responsável por coletar e analisar dados importantes sobre o funcionamento do sistema, permitindo:
- Avaliar eficiência
- Identificar problemas
- Otimizar operação
- Melhorar performance
- Reduzir custos
- Aumentar satisfação
- Facilitar gestão
- Tomar decisões

9. PainelAscii.java

Descrição: Gera a visualização gráfica em ASCII do sistema em funcionamento.

1. Visualização do Sistema
   - Cria representação visual do sistema de elevadores
   - Mostra estado atual dos elevadores
   - Exibe filas de espera
   - Apresenta chamadas ativas

2. Formatação de Interface
   - Gera interface em ASCII
   - Cria bordas e linhas
   - Formata colunas
   - Organiza informações

3. Monitoramento de Elevadores
   - Mostra posição dos elevadores
   - Indica direção de movimento
   - Exibe número de passageiros
   - Representa estado atual

4. Visualização de Filas
   - Mostra pessoas esperando para subir
   - Exibe pessoas esperando para descer
   - Indica prioridades
   - Representa quantidade de pessoas

5. Indicadores de Direção
   - Usa símbolos para direção (^ para subir, v para descer)
   - Mostra estado parado
   - Indica movimento
   - Facilita visualização

6. Representação de Prioridades
   - Marca pessoas prioritárias com *
   - Mostra pessoas comuns com ()
   - Indica quantidade de cada tipo
   - Facilita identificação

7. Formatação de Dados
   - Alinha informações
   - Limita tamanho de texto
   - Organiza colunas
   - Mantém layout consistente

8. Geração de Resumo
   - Cria resumo das filas

- Mostra estatísticas
- Apresenta informações relevantes
- Facilita análise

9. Métodos de Utilidade
- substituirSeta(): Converte símbolos Unicode para ASCII
- gerarResumoFila(): Cria resumo das filas
- repetir(): Repete strings para formatação
- getPainelAsString(): Gera representação completa

10. Layout do Painel
- Cabeçalho com informações
- Colunas para elevadores
- Seção de chamadas
- Bordas e separadores

10. **PainelControle.java**

Descrição: Gerencia os botões de chamada de elevador em cada andar.

Responsabilidades:

- Controlar o estado dos botões “subir” e “descer” por andar.

- Ativar e desativar chamadas conforme acionamento pelos passageiros.

- Manter o estado dos botões sincronizado com as filas e o controlador.

11. **Horario.java**

Descrição: Gerencia o tempo da simulação.

Responsabilidades:

- Controlar o avanço da hora e minuto na simulação.
- Identificar períodos de pico, onde a demanda de passageiros é maior.
- Formatar o horário para exibição e relatórios.

12. **Log.java**

Descrição: Responsável pelo registro de eventos durante a simulação.

Responsabilidades:

- Armazenar mensagens, erros, avisos e acontecimentos importantes.
- Controlar o tamanho do log para evitar uso excessivo de memória.
- Exibir ou salvar o histórico de eventos para análise posterior.
- Considerações Finais
- Este sistema foi concebido para simular de forma realista a operação de elevadores em edifícios altos, considerando:

- Atendimento prioritário a idosos e cadeirantes, promovendo acessibilidade.

- Ajustes dinâmicos para horários de pico, otimizando o fluxo.

- Controle rigoroso do consumo energético e tempos de espera

- Simulação em tempo real com visualização detalhada e logs completos.

Fonte dos dados colocados:

MEGAWHAT. “Com 52% do consumo de eletricidade do país, edificações apresentam maior potencial para eficiência energética”. Disponível em: https://megawhat.energy/mercado-energetico/consumo/com-52-do-consumo-de-eletricidade-do-pais-edificacoes-apresentam-maior-potencial-para-eficiencia-energetica/.

EIXOS INTELIGÊNCIA. “Brasil ficou 14% mais eficiente energeticamente em 13 anos”. Disponível em: https://eixos.com.br/newsletters/brasil-ficou-14-mais-eficiente-energeticamente-em-13-anos/.