Especificação de Integração
MGConnect ADM ↔ MGConnect Albert-View
V3.1

Especificação de Integração — MGConnect ADM ↔ MGConnect Albert-View V3.1

1. Objetivo da integração

O objetivo é permitir que o usuário seja autenticado no MGConnect ADM e, a partir dessa autenticação prévia, seja redirecionado ao MGConnect Albert-View com um JWT assinado, para que o MGConnect Albert-View:

valide o token

crie uma sessão interna própria

abra diretamente a UI atual do Albert-View

aplique escopo por sigla, placa, modo e permissões

mantenha a navegação posterior baseada na sessão, e não mais no token

Esse desenho segue a V3 do protocolo: o JWT é apenas mecanismo de entrada, e a continuidade operacional ocorre por sessão interna no MGConnect Albert-View.

A V3 mantém compatibilidade conceitual com a V2, acrescentando:

versionamento explícito do protocolo (ver)

modos operacionais (mode)

ampliação do catálogo de permissões

integração administrativa entre MGConnect ADM e MGConnect Albert-View

2. O que já foi implementado do lado MGConnect Albert-View

2.1. Entrada autenticada criada

Foi implantada a rota pública de entrada autenticada:

https://albert-view.mgconnect.online/mgconnect/auth/jwt

Ela recebe:

?token=

e executa o fluxo:

recebe o token

valida assinatura e claims

cria sessão interna Flask

redireciona para:

/

ou seja, para a UI atual do Albert-View, sem tela intermediária.

Esse redirecionamento foi ajustado explicitamente de:

/mgconnect/

para:

/

preservando a experiência atual da aplicação.

2.2. Backend MGConnect separado do legado

Foi criado um backend dedicado do MGConnect, rodando separado do Albert View legado:

mgconnect_app.py porta 5571 albertview_api_v2.py porta 5570

O Nginx roteia:

/mgconnect/ → 5571 /view/api/ → 5570 / → UI estática atual do Albert View

Essa separação foi feita sem quebrar a aplicação atual.

2.3. Sessão interna implantada e funcional

O MGConnect cria uma sessão Flask contendo, de forma plana, os campos principais do usuário autenticado.

Campos atuais da sessão:

ver sub name mode siglas siglas_all placas placas_all permissions permissions_all

A sessão é assinada com:

ALBERT_VIEW_SESSION_SECRET

e tanto o:

mgconnect_app.py

quanto o:

albertview_api_v2.py

utilizam a mesma chave de sessão por variável de ambiente, garantindo leitura compartilhada da sessão entre os dois componentes.

Isso foi validado em execução.

2.4. Validação do JWT implantada

A validação do JWT já está funcionando com:

algoritmo HS256 iss = "mgtech360" aud = "mgconnect_albert_view" verificação de assinatura verificação de expiração

Casos de erro já retornam JSON consistente, por exemplo:

token ausente token inválido falha de assinatura issuer inválido audience inválida token expirado

Esse comportamento foi testado e validado.

2.5. Nginx e stack operacional implantados

O ambiente atual está operacional com:

mgconnect_app.py em 5571 albertview_api_v2.py em 5570 Nginx ativo em 443 /mgconnect/ roteado corretamente /view/api/ roteado corretamente

O:

albert_stack.sh

foi atualizado para iniciar, parar e registrar logs do MGConnect, além de exportar:

MGCONNECT_JWT_SECRET ALBERT_VIEW_SESSION_SECRET

Esse stack foi validado em execução.

2.6. Escopo por sigla e placa já aplicado à UI e ao backend

No backend legado

Os endpoints relevantes da View passaram a ler a sessão e filtrar dados por:

```
siglas_all siglas placas_all placas
```

Esse filtro já afeta:

seletores cards/minicards mapa

No frontend legado

A UI foi adaptada para:

consultar a sessão via /mgconnect/api/me montar os seletores de sigla e placa conforme o escopo permitido mostrar a opção "Todas" quando houver mais de uma opção disponível

Esse comportamento foi testado com perfis restritos e perfis amplos.

2.7. Permissões visuais dos botões já aplicadas

Foi implantada no frontend a lógica de permissões dos botões do modal.

Mapeamento atual:

view_dashboard → painel de instrumentos view_charts → gráficos view_routes → jornadas & rotas view_diagnostics → diagnóstico do ciclo view_predictive → predição ML view_alerts → eventos & notificações view_summary → resumo view_driver → pagina do motorista

Regras atuais:

permissions_all = true → libera tudo permissions_all = false → exibe somente o que estiver em permissions sem sessão → comportamento legado preservado

O enforcement de backend para endpoints sensíveis permanece como próxima etapa.

2.8. Modos operacionais

A V3 introduz o conceito de modo operacional.

Campo:

"mode": "max_agent"

Valores válidos:

super super_agent admin admin_agent pro pro_agent max max_agent driver

super

Usuário interno MGConnect.

Possui acesso completo de visualização.

Não possui acesso de edição.

super_agent

Mesmo comportamento de:

super

com acesso adicional ao agente.

admin

Usuário interno MGConnect.

Possui acesso completo ao sistema.

Possui acesso às funções de edição autorizadas do MGConnect Albert-View.

admin_agent

Mesmo comportamento de:

admin

com acesso adicional ao agente.

pro

Possui acesso:

página inicial cards minicards seletores mapa

Ao clicar em um veículo:

abre o mapa

Não abre o modal do veículo.

pro_agent

Mesmo comportamento de:

pro

com acesso adicional ao agente.

max

Possui acesso ao modal do veículo.

Os módulos internos disponíveis no modal são definidos por:

```
permissions permissions_all
```

max_agent

Mesmo comportamento de:

max

com acesso adicional ao agente.

driver

Acesso exclusivo à:

Página do Motorista / Gamificação

Não participa da navegação tradicional do Albert-View.

2.9. Fluxo funcional já validado

O fluxo completo já foi validado com tokens de teste:

1. abrir /mgconnect/auth/jwt?token=... 2. validar o token 3. criar sessão 4. redirecionar para / 5. abrir o Albert-View 6. aplicar sessão no frontend e backend

Esse fluxo foi validado inclusive em navegador.

O estado atual da stack foi reportado como operacional.

3. O que falta do lado MGConnect Albert-View

3.1. Validar escopo recebido contra a base real

Hoje o token é validado criptograficamente e a sessão é criada.

A próxima melhoria recomendada é validar operacionalmente:

se as siglas existem se as placas existem se a placa pertence ao universo esperado se as permissões recebidas são válidas se o usuário está habilitado

Se falhar:

não criar sessão retornar erro apropriado

3.2. Proteger endpoints sensíveis no backend

A UI já oculta funcionalidades por permissão.

O backend ainda deverá bloquear endpoints como:

ML / predição diagnósticos alertas preferências de alerta agente

3.3. Evoluir o domínio de dados

Itens como:

motorista ativo rankings análises cruzadas gamificação

devem ser tratados como dados persistidos no banco e não como claims JWT.

3.4. Integração Administrativa MGConnect ADM ↔ MGConnect Albert-View

Além do protocolo JWT, a V3 estabelece integração administrativa entre:

MGConnect ADM ↕ Banco Compartilhado ↕ MGConnect Albert-View

Essa integração é independente do fluxo JWT.

Seu objetivo é compartilhar informações administrativas e operacionais entre as duas plataformas.

3.4.1. Campos compartilhados

Os campos compartilhados são:

status_cmd num_serie sigla empresa placa marca modelo geral

conjuntos de dados: sigla - número de registro do motorista – nome do motorista – placa do veículo vinculado no momento (se houver) – num_serie

3.4.2. status_cmd

O campo:

status_cmd

possui os seguintes valores:

0 = liberado 1 = deprecado somente GPS e informações básicas 2 = bloqueado

3.4.3. Sincronização

Deve existir mecanismo de sinalização de alterações entre:

MGConnect ADM MGConnect Albert-View

para evitar varreduras contínuas no banco compartilhado.

A implementação fica em aberto e poderá utilizar:

fila eventos flags notificações ou mecanismo equivalente

4. O que o MGConnect ADM deve implementar

Agora a parte mais importante para a equipe responsável pelo MGConnect ADM.

A V3 mantém o mesmo modelo arquitetural da V2:

o MGConnect ADM autentica o usuário

o MGConnect ADM resolve o escopo

o MGConnect ADM define o modo operacional

o MGConnect ADM define as permissões

o MGConnect ADM gera o JWT

o MGConnect ADM redireciona o navegador para o Albert-View

O Albert-View não realiza autenticação própria.

Ele apenas valida o token recebido e cria sua sessão interna.

4.1. Papel do MGConnect ADM

O MGConnect ADM é responsável por:

autenticar o usuário

decidir se ele pode acessar o módulo MGConnect Albert-View

montar o escopo do usuário

definir o modo operacional

definir as permissões

gerar o JWT assinado

redirecionar o navegador para o Albert-View

Esse papel continua sendo o mesmo da V2.

4.2. Endpoint de destino

O MGConnect ADM deve redirecionar o navegador para:

https://albert-view.mgconnect.online/mgconnect/auth/jwt?token=

Esse é o endpoint oficial de entrada autenticada.

4.3. Header do JWT

O cabeçalho do JWT deve ser:

{

"alg": "HS256",

"typ": "JWT"

}

A V3 continua utilizando:

HS256

como algoritmo oficial.

4.4. Payload do JWT

O payload esperado pela V3 é:

{

"ver": 3,

"sub": "usr_1",

"name": "Admin",

"mode": "max_agent",

"siglas": [],

"siglas_all": true,

"placas": [],

"placas_all": true,

"permissions": [],

"permissions_all": true,

"iat": 1774855200,

"exp": 1774855380,

"iss": "mgtech360",

"aud": "mgconnect_albert_view"

}

Campos obrigatórios

`ver`

Versão do protocolo.

Na V3:

"ver": 3

Valor obrigatório.

`sub`

Identificador estável do usuário.

Exemplo:

"sub": "usr_123"

Não deve mudar entre logins.

`name`

Nome de exibição.

Exemplo:

"name": "Administrador"

Utilizado para apresentação.

`mode`

Define o modo operacional do usuário.

Exemplo:

"mode": "max_agent"

Valores válidos:

super

super_agent

admin

admin_agent

pro

pro_agent

max

max_agent

driver

`siglas`

Lista explícita de siglas autorizadas.

Exemplo:

"siglas": [

"MGT",

"SPO"

]

`siglas_all`

Controle de acesso global às siglas.

Exemplo:

"siglas_all": true

`placas`

Lista explícita de placas autorizadas.

Exemplo:

"placas": [

"ABC1234",

"DEF5678"

]

`placas_all`

Controle de acesso global às placas.

Exemplo:

"placas_all": true

`permissions`

Lista explícita de permissões.

Exemplo:

"permissions": [

"view_routes",

"view_predictive"

]

`permissions_all`

Controle de acesso global às permissões.

Exemplo:

"permissions_all": true

`iat`

Timestamp de emissão.

`exp`

Timestamp de expiração.

`iss`

Deve ser exatamente:

"iss": "mgtech360"

Observação: embora o sistema passe a ser denominado MGConnect ADM, o identificador técnico do emissor permanece "mgtech360" para preservar compatibilidade com a infraestrutura já implantada.

`aud`

Deve ser exatamente:

"aud": "mgconnect_albert_view"

4.5. Segredo compartilhado

O MGConnect ADM deve utilizar exatamente o mesmo:

MGCONNECT_JWT_SECRET

utilizado pelo MGConnect Albert-View.

Regras:

não hardcodar

usar variável de ambiente

chave longa

chave aleatória

mesma chave nos dois lados

Esse requisito é obrigatório.

4.6. Regras de escopo

Siglas

Se o usuário possui acesso total:

{

"siglas_all": true,

"siglas": []

}

Se possui acesso restrito:

{

"siglas_all": false,
"siglas": [
"MGT",
"SPO"
]
}

Placas

Se o usuário possui acesso total:

{
"placas_all": true,
"placas": []
}

Se possui acesso restrito:

{
"placas_all": false,
"placas": [
"ABC1234",
"DEF5678"
]
}

Permissões

Se possui acesso total:

{
"permissions_all": true,
"permissions": []
}

Se possui acesso parcial:

{
"permissions_all": false,
"permissions": [
"view_routes",
"view_predictive"
]
}

4.7. Catálogo oficial de permissões

O conjunto oficial de permissões reconhecidas pelo MGConnect Albert-View V3 é:

```
view_dashboard
view_charts
view_routes
view_diagnostics
view_predictive
view_alerts
view_summary
view_driving
```

O MGConnect ADM deve emitir exatamente esses nomes.

Não devem existir variações.

Não devem existir aliases.

Não devem existir abreviações.

4.8. Regras dos modos operacionais

super

Acesso completo de visualização.

Sem edição.

super_agent

Mesmo comportamento do super com acesso ao agente.

admin

Acesso completo.

Possui acesso às funcionalidades de edição autorizadas do MGConnect Albert-View.

admin_agent

Mesmo comportamento do admin com acesso ao agente.

pro

Possui acesso:

página inicial

cards

minicards

mapa

Ao clicar em um veículo:

abre mapa

Não abre modal.

pro_agent

Mesmo comportamento do pro com acesso ao agente.

max

Possui acesso ao modal do veículo.

Os módulos disponíveis dentro do modal são definidos pelas permissões.

max_agent

Mesmo comportamento do max com acesso ao agente.

driver

Abre diretamente:

Página do Motorista / Gamificação

Sem acesso à navegação tradicional do sistema.

4.9. Exemplos de perfis

Super usuário MGConnect

{
"ver": 3,
"sub": "mgc_admin",
"name": "Administrador MGConnect",
"mode": "super",
"siglas_all": true,
"siglas": [],
"placas_all": true,
"placas": [],
"permissions_all": true,
"permissions": []
}

Proprietário de frota

{
"ver": 3,
"sub": "owner_1",
"name": "Proprietário",
"mode": "max",
"siglas_all": false,
"siglas": ["MGT"],
"placas_all": true,

"placas": [],

"permissions_all": false,

"permissions": [

"view_routes",

"view_predictive",

"view_summary",

"view_driving"

]

}

Mecânico

{

"ver": 3,

"sub": "mec_1",

"name": "Mecânico",

"mode": "max",

"permissions_all": true

}

Motorista

{

"ver": 3,

"sub": "drv_1",

"name": "Motorista",

"mode": "driver",

"permissions_all": false,

"permissions": []

}

4.10. Exemplo de geração no MGConnect ADM

```
const jwt = require("jsonwebtoken");

const SECRET =
process.env.MGCONNECT_JWT_SECRET;

function gerarTokenAlbertView(usuario) {

const agora =
```

```
Math.floor(Date.now() / 1000);

const payload = {
      ver: 3,
      sub: usuario.id,
      name: usuario.name,
      mode: usuario.mode,
      siglas: usuario.siglas || [],
      siglas_all: !!usuario.siglas_all,
      placas: usuario.placas || [],
      placas_all: !!usuario.placas_all,
permissions: usuario.permissions || [],
permissions_all: !!usuario.permissions_all,
      iat: agora,
      exp: agora + 120,
      iss: "mgtech360",
      aud: "mgconnect_albert_view"
};

return jwt.sign(
payload,
SECRET,
{
algorithm: "HS256"
}
);
}
```

Esse formato deriva diretamente do protocolo V3.

5. Regras de erro que o MGConnect ADM deve considerar

Se o MGConnect ADM emitir um token inconsistente, o lado MGConnect Albert-View poderá responder com erros padronizados para facilitar diagnóstico e correção.

Os códigos de retorno previstos são:

400 — Requisição inválida

Utilizado para situações como:

token ausente

payload inválido

parâmetros obrigatórios ausentes

formato incorreto da requisição

Exemplos:

token não informado

payload malformado

401 — Não autorizado

Utilizado para situações como:

token inválido

assinatura inválida

token expirado

```
issuer incorreto
audience incorreta
```

Exemplos:

assinatura JWT inválida

token expirado

```
iss incompatível
aud incompatível
```

403 — Acesso negado

Utilizado para situações como:

permissão inexistente

modo operacional incompatível

acesso não autorizado ao recurso solicitado

tentativa de utilização de funcionalidade não permitida

Exemplos:

usuário sem permissão para o módulo

modo operacional incompatível

422 — Inconsistência semântica

Utilizado para situações como:

sigla inexistente

placa inexistente

permissão desconhecida

associação inválida entre escopo e recurso

inconsistência operacional detectada durante validação

Exemplos:

placa não pertence à sigla

sigla inexistente

permissão desconhecida

500 — Erro interno

Utilizado quando ocorrer falha interna inesperada do lado MGConnect Albert-View.

Exemplos:

erro interno de processamento

falha de infraestrutura

falha de banco de dados

erro não tratado

Esses códigos devem ser utilizados pela equipe do MGConnect ADM para diagnóstico rápido de problemas de integração.

6. Contrato funcional final entre as duas pontas

A V3 estabelece claramente a separação de responsabilidades entre os componentes.

MGConnect ADM faz

O MGConnect ADM é responsável por:

autenticar o usuário

validar credenciais

resolver o escopo operacional

definir siglas autorizadas

definir placas autorizadas

definir permissões

definir modo operacional

gerar o JWT

assinar o JWT

redirecionar o navegador para o endpoint do Albert-View

O MGConnect ADM é a única autoridade de autenticação do protocolo.

MGConnect Albert-View faz

O MGConnect Albert-View é responsável por:

validar o JWT recebido

validar assinatura

validar issuer

validar audience

validar expiração

criar sessão interna

abrir a UI atual

aplicar escopo por sigla

aplicar escopo por placa

aplicar modo operacional

aplicar permissões de interface

aplicar permissões de backend

controlar a navegação posterior através da sessão

O Albert-View não autentica usuários diretamente.

Banco Compartilhado

A V3 acrescenta uma segunda camada de integração independente do JWT.

Fluxo:

MGConnect ADM

↕

Banco Compartilhado

↕

MGConnect Albert-View

Essa integração é destinada a informações administrativas e operacionais compartilhadas.

Campos compartilhados

Os campos oficiais compartilhados são:

status_cmd

num_serie

sigla

empresa

placa

marca

modelo

geral

Controle operacional

O campo:

status_cmd

define o estado administrativo do ativo.

Valores:

0 = liberado

1 = deprecado

somente GPS e informações básicas

2 = bloqueado

Sincronização

Deve existir mecanismo de sinalização de alterações entre:

MGConnect ADM

MGConnect Albert-View

para evitar consultas contínuas ao banco compartilhado.

A implementação poderá utilizar:

filas

eventos

flags

notificações

mecanismos equivalentes

sem restrição imposta por este protocolo.

7. Resumo executivo

Do lado MGConnect Albert-View

Já está implantado:

endpoint /mgconnect/auth/jwt

validação JWT

validação de assinatura

validação de expiração

validação de issuer

validação de audience

sessão compartilhada com a View atual

abertura da UI sem tela intermediária

filtros por sigla

filtros por placa

permissões visuais

infraestrutura Nginx

stack operacional

logs operacionais

Também já se encontra incorporado ao protocolo V3:

suporte ao campo ver

suporte ao campo mode

suporte às novas permissões

suporte aos modos operacionais

Do lado MGConnect ADM

Deve ser implementado:

autenticação dos usuários

resolução de escopo

definição de permissões

definição de modo operacional

geração do JWT HS256

uso do segredo compartilhado

emissão dos claims obrigatórios

redirecionamento para o endpoint do Albert-View

Claims obrigatórios da V3

O JWT deve conter:

```
ver
sub
name
mode
siglas
siglas_all
placas
placas_all
permissions
permissions_all
iat
exp
iss
aud
```

Modos operacionais da V3

Os modos reconhecidos são:

super

super_agent

admin

admin_agent

pro

pro_agent

max

max_agent

driver

Permissões reconhecidas pela V3

```
view_dashboard
view_charts
view_routes
view_diagnostics
view_predictive
view_alerts
view_summary
view_driving
```

Integração administrativa

A V3 acrescenta integração administrativa entre:

MGConnect ADM

MGConnect Albert-View

através de banco compartilhado.

Campos oficiais:

status_cmd

num_serie

sigla

empresa

placa

marca

modelo

geral

Resumo final

A V3 preserva integralmente os conceitos arquiteturais da V2 e amplia o protocolo com:

versionamento explícito do JWT

modos operacionais

novas permissões

separação formal entre autenticação e operação

integração administrativa entre plataformas

compartilhamento de estado operacional através de banco comum

O JWT continua sendo exclusivamente um mecanismo de entrada.

A continuidade operacional permanece baseada em sessão interna do MGConnect Albert-View.