🔗 Guia de Integração Frontend-Backend UCONNECT

📋 Índice

🔧 Alterações no Backend

1. Corrigir o modelo Event

O schema `EventResponse` tem um campo `eventType` que não existe no modelo. Adicione ao `models.py`:

python

```python
# Em models.py, na classe Event
class Event(Base):
    __tablename__ = "Event"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String(200), nullable=False)
    description = Column(Text, nullable=True)
    timestamp = Column(DateTime, nullable=False)
    academicGroupId = Column(String(50), nullable=True)
    eventType = Column(String(50), nullable=True, default="evento-geral")  # ADICIONAR ESTA LINHA

    creatorId = Column(Integer, ForeignKey("User.id", ondelete="SET NULL"), nullable=True)

    creator = relationship("User", back_populates="events_created")

    __table_args__ = (
        Index("idx_event_timestamp", "timestamp"),
        Index("idx_event_creator", "creatorId"),
    )
```

2. Atualizar dependencies.py

Há uma duplicação. O correto é usar apenas uma das versões. Recomendo usar a versão em `dependencies.py` e remover de `utils.py`:

python

```python
# Em utils.py, REMOVER as funções get_current_user, get_current_active_user e require_roles
# Elas devem existir APENAS em dependencies.py
```

3. Corrigir inconsistência no token

Em `utils.py`, o token está salvando `registration` mas em `dependencies.py` está buscando por `id`. Corrija:

python

```python
# Em dependencies.py - linha ~24
async def get_current_user(token: str = Depends(oauth2_scheme), db: Session = Depends(get_db)) -> models.User:
    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Não foi possível validar as credenciais",
        headers={"WWW-Authenticate": "Bearer"},
    )

    payload = decode_token(token)
    if payload is None:
        raise credentials_exception

    # MUDAR de "id" para "sub" e buscar por registration
    registration: str = payload.get("sub")  # Mudança aqui
    if registration is None:
        raise credentials_exception

    # Buscar por registration, não por id
    user = db.query(models.User).filter(models.User.registration == registration).first()
    if user is None:
        raise credentials_exception

    return user
```

4. Corrigir imports nos routers

Em `users.py` e `events.py`, corrija os imports:

python

```
# Em users.py
from ..dependencies import get_current_active_user, require_roles
# REMOVER: from .. import utils (se estiver usando utils.require_roles)

# Em events.py
from ..dependencies import get_current_user, require_roles
# Já está correto!
```

5. Migração do Banco de Dados

Se já tem dados, você precisa adicionar a coluna `eventType`:

sql

```
-- Execute no seu banco de dados
ALTER TABLE Event ADD COLUMN eventType VARCHAR(50) DEFAULT 'evento-geral';
```

Ou recrie o banco (se estiver em desenvolvimento):

bash

```
# Apague o arquivo do banco SQLite e recrie
rm instance/database.db  # ou o caminho do seu DB
python -c "from app.db import Base, engine; Base.metadata.create_all(bind=engine)"
```

📁 Estrutura do Frontend

Organize seus arquivos assim:

```
frontend/
├── public/
│   ├── index.html
│   └── ...
├── src/
│   ├── components/
│   │   ├── Calendario.jsx      (arquivo atualizado)
│   │   ├── Login.jsx           (novo arquivo)
│   │   └── styles/
│   │       ├── calendario.css
│   │       └── login.css       (novo arquivo)
│   ├── services/
│   │   └── api.js              (novo arquivo)
│   ├── App.js                  (atualizado)
│   └── index.js
├── .env                        (novo arquivo)
└── package.json
```

Arquivos para criar/atualizar:

1. **src/services/api.js** - Novo (já fornecido)
2. **src/components/Login.jsx** - Novo (já fornecido)
3. **src/components/Calendario.jsx** - Substituir o components.jsx
4. **src/components/styles/login.css** - Novo (já fornecido)
5. **src/App.js** - Atualizar (já fornecido)
6. **.env** - Criar na raiz do projeto (já fornecido)

🚀 Instalação e Execução

Backend (FastAPI)

bash

```bash
# No diretório backend/
cd backend

# Ativar ambiente virtual
source venv/bin/activate  # Linux/Mac
# ou
venv\Scripts\activate     # Windows

# Instalar dependências (se ainda não fez)
pip install fastapi sqlalchemy python-jose passlib bcrypt python-multipart

# Executar o servidor
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

O backend estará rodando em: `http://localhost:8000`

Frontend (React)

bash

```bash
# No diretório frontend/
cd frontend

# Instalar dependências
npm install

# Criar o arquivo .env (se ainda não criou)
echo "REACT_APP_API_URL=http://localhost:8000" > .env

# Executar o app
npm start
```

O frontend estará rodando em: `http://localhost:3000`

🧪 Testando a Integração

1. Criar um usuário admin (via Python)

python

```
# No diretório backend/, execute:
python

# Dentro do console Python:
from app.db import SessionLocal
from app.models import User, AccessStatus, UserRole
from app.utils import get_password_hash

db = SessionLocal()

admin = User(
    registration="admin001",
    name="Administrador",
    email="admin@uconnect.com",
    passwordHash=get_password_hash("senha123"),
    role=UserRole.admin,
    accessStatus=AccessStatus.active
)

db.add(admin)
db.commit()
print("Usuário admin criado!")
exit()
```

2. Testar o Login

1. Acesse `http://localhost:3000`
2. Use as credenciais:
    - **Matrícula:** `admin001`
    - **Senha:** `senha123`

3. Testar CRUD de Eventos

Com o usuário admin logado:

1. ✅ **Criar evento:** Clique em "+ Adicionar Evento"
2. ✅ **Visualizar:** Veja o evento no calendário
3. ✅ **Editar:** Clique no evento e altere
4. ✅ **Excluir:** No modal de edição, clique em "Excluir"

🔍 Resolução de Problemas

Erro: "CORS policy"

Sintoma: Console mostra erro de CORS

Solução: Verifique se o CORS está configurado no `main.py` para incluir `http://localhost:3000`

python

```
# Em main.py
origins = {
    "http://localhost:3000",  # Adicionar esta linha
    "http://127.0.0.1:3000",
    # ... outras origens
}
```

Erro: "401 Unauthorized"

Sintoma: Não consegue criar/editar eventos após login

Solução:

1. Verifique se o token está sendo salvo no localStorage (F12 > Application > Local Storage)
2. Verifique se o backend está validando o token corretamente
3. Confira se a correção do `dependencies.py` foi aplicada

Erro: "404 Not Found" ao buscar eventos

Sintoma: Frontend não carrega eventos

Solução: Verifique se o backend está rodando e se a rota é `/events` (não `/eventos`)

Eventos não aparecem no calendário

Sintoma: Login funciona, mas calendário está vazio

Possíveis causas:

1. Não há eventos no banco de dados
2. Formato de data incompatível
3. Erro na conversão `eventFromBackend`

Debug:

javascript

```jsx
// Em Calendario.jsx, adicione console.log
useEffect(() => {
  const fetchEventos = async () => {
    try {
      const data = await getEvents();
      console.log("Eventos recebidos:", data); // Debug
      setEventos(data);
    } catch (err) {
      console.error("Erro:", err);
    }
  };
  fetchEventos();
}, []);
```

Erro: "Cannot read property 'timestamp'"

Sintoma: Erro ao converter evento

Solução: O backend não está retornando o campo `timestamp`. Verifique:

1. Se o modelo `Event` tem o campo `timestamp`
2. Se o schema `EventResponse` inclui `timestamp`
3. Se a migração do banco foi executada

📊 Fluxo de Dados

```
Frontend (React)          Backend (FastAPI)
    |                          |
    |-- POST /auth/login ----------->|
    |<-- token + expires_at -------|
    |                          |
    | [Salva token no localStorage]|
    |                          |
    |-- GET /events ---------------->|
    |   (sem auth - público)       |
    |<-- lista de eventos ---------|
    |                          |
    |-- POST /events -------------->|
    |   (com Authorization header) |
    |   Bearer <token>             |
    |<-- evento criado ------------|
```

🎯 Próximos Passos

1. **Melhorias de UX:**
   - Loading states melhores
   - Feedback de sucesso/erro
   - Confirmações visuais
2. **Funcionalidades:**
   - Filtro por tipo de evento
   - Busca de eventos
   - Exportar calendário
3. **Segurança:**
   - Refresh token
   - Logout automático após expiração
   - Validação de formulários mais robusta
4. **Deploy:**
   - Configurar variáveis de ambiente para produção
   - Build otimizado do React
   - HTTPS

📝 Checklist Final

Antes de considerar a integração completa:

- [ ] Backend rodando sem erros
- [ ] Modelo Event com campo `eventType` (se necessário)
- [ ] Frontend rodando sem erros
- [ ] Arquivo .env configurado
- [ ] Usuário admin criado
- [ ] Login funcionando
- [ ] Listagem de eventos funcionando
- [ ] Criação de eventos funcionando
- [ ] Edição de eventos funcionando

- [ ] Exclusão de eventos funcionando
- [ ] Logout funcionando
- [ ] Permissões sendo respeitadas

Dúvidas? Revise cada passo cuidadosamente. A maioria dos problemas vem de:

1. Endpoints incorretos
2. Token não sendo enviado
3. CORS mal configurado
4. Modelo do banco desatualizado