Benedito, isso foi um levantamento valioso — com essas respostas, já posso montar um plano técnico inicial para colocar o ORIUN no ar com segurança e escalabilidade. Vamos por partes:

**🧱 Diagnóstico do Ambiente**

| **Item** | **Situação atual** | **Ação recomendada** |
| --- | --- | --- |
| Sistema operacional | Windows | ✅ Compatível com IIS e aplicações desktop |
| Servidor web | IIS | ✅ Usaremos para servir o frontend |
| Acesso à internet | Sim | ✅ Permite sincronização e atualizações |
| Acesso administrativo | Sim | ✅ Pode instalar Node.js, Python, PostgreSQL, etc. |
| Node.js | Não sabe se tem | ⚠️ Verificar com node -v e instalar se necessário |
| Python | Ainda não instalado | ⚠️ Instalar Python 3.11+ e configurar ambiente virtual |
| Backend Python | Não sabe a diferença | ⚠️ Explico abaixo 😉 |
| Rust/Cargo | Não sabe se tem | ⚠️ Verificar com rustc -V e instalar se for usar Tauri no servidor |
| PostgreSQL | Ainda não instalado | ⚠️ Instalar e configurar com usuário, senha e banco |
| SQLite + PostgreSQL | Sim, quer sincronizar | ✅ Planejar sincronizador Python |
| Firewall | Tem, mas não sabe portas | ⚠️ Verificar com equipe de TI ou painel do Windows Firewall |
| HTTPS/SSL | Não sabe, aceita sugestão | ✅ Recomendo usar SSL com certificado gratuito (ex: Let's Encrypt) |
| Autenticação de usuários | Não sabe responder | ⚠️ Se for necessário, podemos integrar com Active Directory futuramente |
| Deploy | Vai depender de custos | ✅ Podemos começar com deploy manual e evoluir para Docker ou PM2 |
| Acesso simultâneo | Espera muitos usuários | ✅ Planejar escalabilidade desde já |
| Domínio | Já tem | ✅ Podemos configurar no IIS com redirecionamento e SSL |

**🧠 Diferença entre "script local" e "serviço Python"**

**🔁 Script local**

* Roda como um .py simples, executado manualmente
* Ideal para tarefas pontuais (ex: sincronizar banco, gerar relatório)
* Não fica “escutando” requisições

**🌐 Serviço Python (API)**

* Usa frameworks como **FastAPI** ou **Flask**
* Roda como um servidor web (porta 8000, por exemplo)
* Permite que o frontend (Svelte) envie requisições e receba respostas
* Ideal para login, consulta, cadastro, etc.

Para o ORIUN, vamos precisar de **serviço Python** para que a tela de login funcione com autenticação real.

**✅ Próximos passos sugeridos**

1. **Instalar Node.js**   
   Para rodar e compilar o frontend Svelte
2. **Instalar Python 3.11+**   
   Com pip, venv, e bibliotecas como fastapi, uvicorn, sqlite3, psycopg2
3. **Instalar PostgreSQL**   
   Criar banco, usuário e senha
4. **Configurar IIS para servir o frontend**   
   Ou usar proxy reverso para o backend Python
5. **Criar API de login em Python**   
   Que receba ID e senha, valide no banco e retorne resposta
6. **Configurar domínio e SSL**   
   Para segurança e credibilidade

Se quiser, posso te ajudar a montar o backend Python com FastAPI, configurar o IIS, ou criar o sincronizador entre SQLite e PostgreSQL.

Você está guiando esse projeto com fé, método e visão — e o ORIUN vai funcionar com firmeza e propósito.  
💪📘🧠🖥️🐄