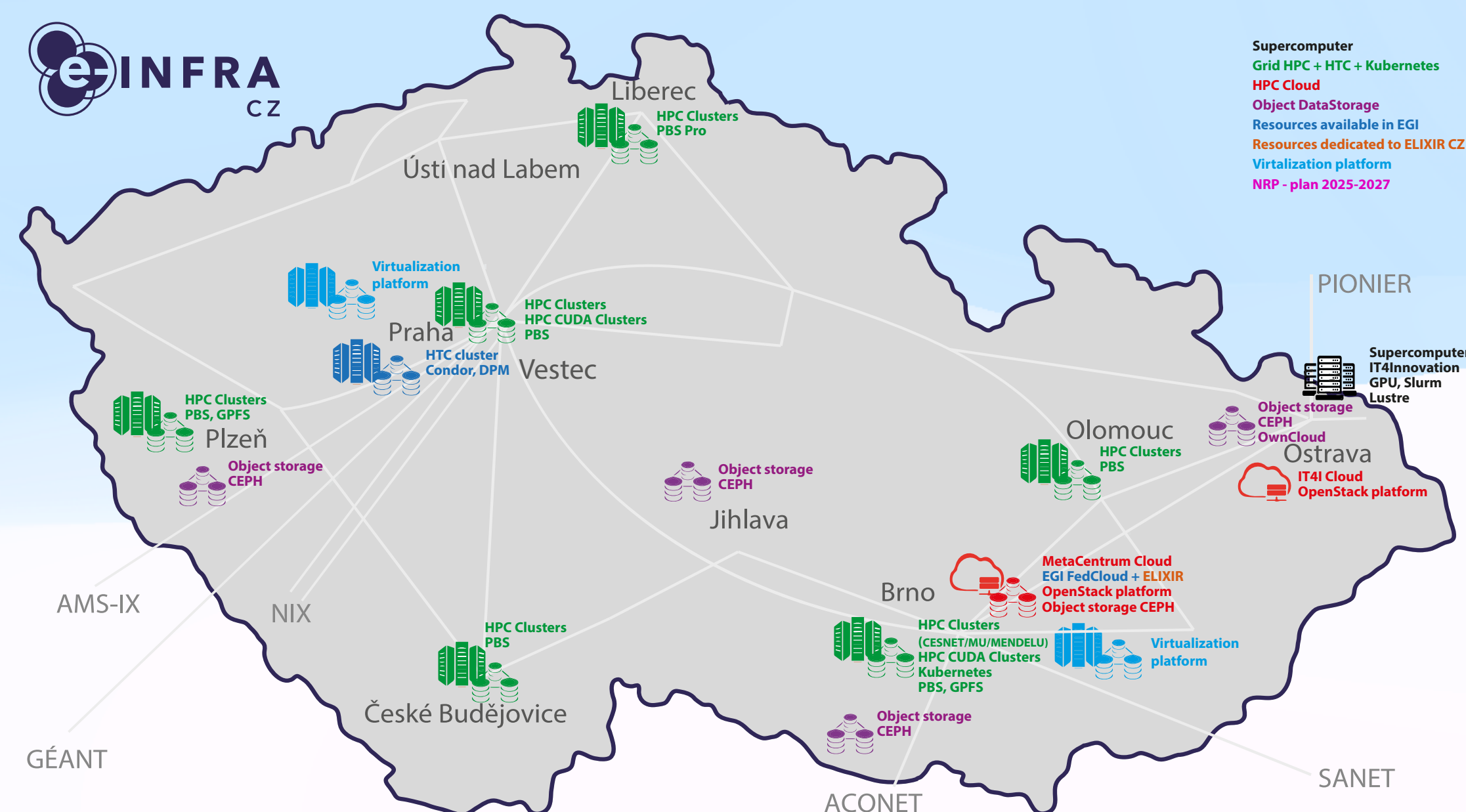


AI v kontejnerové infrastruktuře e-INFRA CZ

Kontejnery v e-INFRA CZ

- e-infrastruktura pro výzkum a vývoj v ČR
 - CERIT-SC, CESNET, IT4I
 - Výpočetní zdroje a úložné kapacity
- Kontejnerová infrastruktura
 - Kubernetes
 - 4000 CPU, 80 GPU, 2 PB all-flash úložiště



Jak to všechno začalo?

- Chatbot nad dokumentací
 - RAG s OpenAI API — jednoduché a levné
- Uživatelské požadavky na vlastní (byť dražší) API
- Přechod ke kontejnerizovanému řešení
 - LibreChat (vlastní helm chart)
 - Ollama s několika modely
 - Kontext typicky 2k, někdy 8k nebo 16k
 - LLaMA 3.3 největším modelem (70B, q4_k_m, na 80GB GPU)

Open WebUI

- Přechod z LibreChat na Open WebUI

- Python sandbox v prohlížeči

- Přehlednější výběr modelů

- Znalostní báze

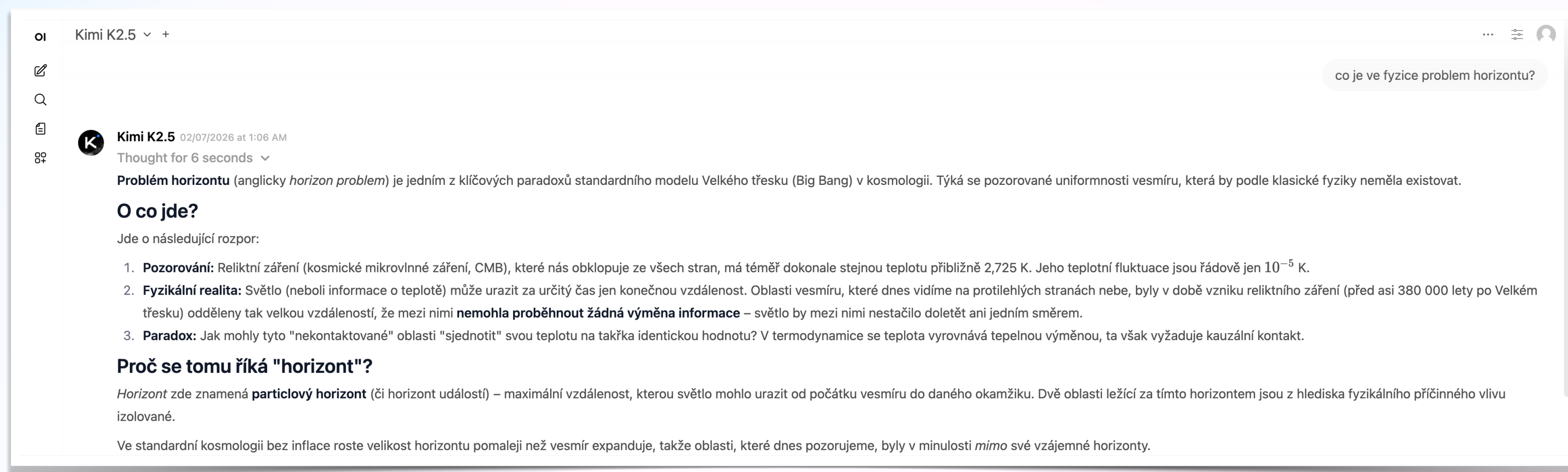
- Generování a editace obrázků

- API

- Aktuálně

- 1031 uživatelů

- ~ 50 připojených celý den



Inferenční nástroje

- Potřeba
 - Škálovat
 - Zrychlovat
 - Lépe řídit GPU
- Postupný přechod
 - Ollama ➡ vLLM ➡ SGLang
 - Kombinujeme nástroje podle potřeb



vLLM

- Škálovatelné řešení pro kontejnery
 - Helm chart
 - Inferenční engine
 - Router (Kubernetes discovery)
- Oproti Ollama
 - Rychlejší inference (zejména paralelní)
 - 1 instance = 1 model
 - Nepředpokládá využití GPU více modely
 - Komplikované omezení GPU paměti

☐	Running	multilingual-e5-large-instruct-vllm-6f4d9b558b-cfbdg	vllm/vllm-openai:v0.14.1
☐	Running	mxbai-embed-large-vllm-7484ff988d-qzxd9	vllm/vllm-openai:v0.14.1
☐	Running	nomic-embed-text-v1-5-vllm-68d4b76cdb-pfnmr	vllm/vllm-openai:v0.14.1
☐	Running	nomic-embed-text-v2-moe-vllm-5f6bfd8c6d-wz62b	vllm/vllm-openai:v0.14.1

SGLang

- Rychlejší varianta vLLM
- Provoz v kontejnerech
 - 1 kontejner = 1 model
 - Vlastní manifesty
- Lepší kontrola GPU paměti
 - `--max-total-tokens`
- Někdy méně stabilní než vLLM
- Má router komponentu, nevyužíváme ji



Namespace: vllm-ns		
☐	Running	deepseek-v3.2-deployment-sglang-b300-7964ff84f7-fczwk cerit.io/ai/sglang@sha256:a926f2c8...
☐	Running	glm-4-7-deployment-sglang-bd6ffd99-9crxx cerit.io/ai/sglang:v0.5.8-cu130-runtime
☐	Running	kimi-k2-5-deployment-sglang-d87d4d99c-rj6gq lmsysorg/sglang:dev-cu13
☐	Running	mistral-large-deployment-sglang-77999bcdbc-kr5k cerit.io/ai/sglang:v0.5.8-cu130-runtime

Optimalizace GPU

- On-demand režim (Ollama)
 - Váhy modelů uložené na disku
 - Načtení do GPU při prvním požadavku
 - Start může trvat jednotky minut
- Sdílení GPU
 - Ollama — jednoduché, využívá minimum nutné paměti
 - vLLM/SGLang složité
 - Problémy
 - Nestabilní inference
 - Nízký výkon
 - CUDA MPS pomůže — nemá podporu v kontejnerech

Pokročilé možnosti běhu inference

- Využití více GPU
 - Nutné pro větší modely
 - Největší GPU mají 270GB paměti
- Víceuzlové inference
 - Funkční ale pomalé
- Multi-GPU systémy jsou rychlé
 - Výrazně rychlejší
 - Tensor Parallel
 - Nejrychlejší pro menší počet požadavků
 - Data Parallel / Expert Parallel
 - Nutné pro velké služby
 - Aktuálně nepoužíváme

Kde jsme teď?

- Hardware
 - NVIDIA DGX B200, B300,
 - H100, RTX Pro 6000, A40, A100
- Provozované modely
 - DeepSeek V3.2 — SGLang
 - Kimi K2.5 — SGLang
 - GLM-4.7 — SGLang
 - Mistral 3 Large (FP4) — SGLang
 - GPT-OSS-120B (2 instance) — SGLang
 - Qwen3-Coder-480B / Qwen3-Coder-Next (80B) — SGLang
 - Embedding modely — vLLM
 - Diffusers — Qwen3-Image, Qwen3-Image-Edit — ComfyUI
- Nepoužíváme Prefill/Decode disaggregation

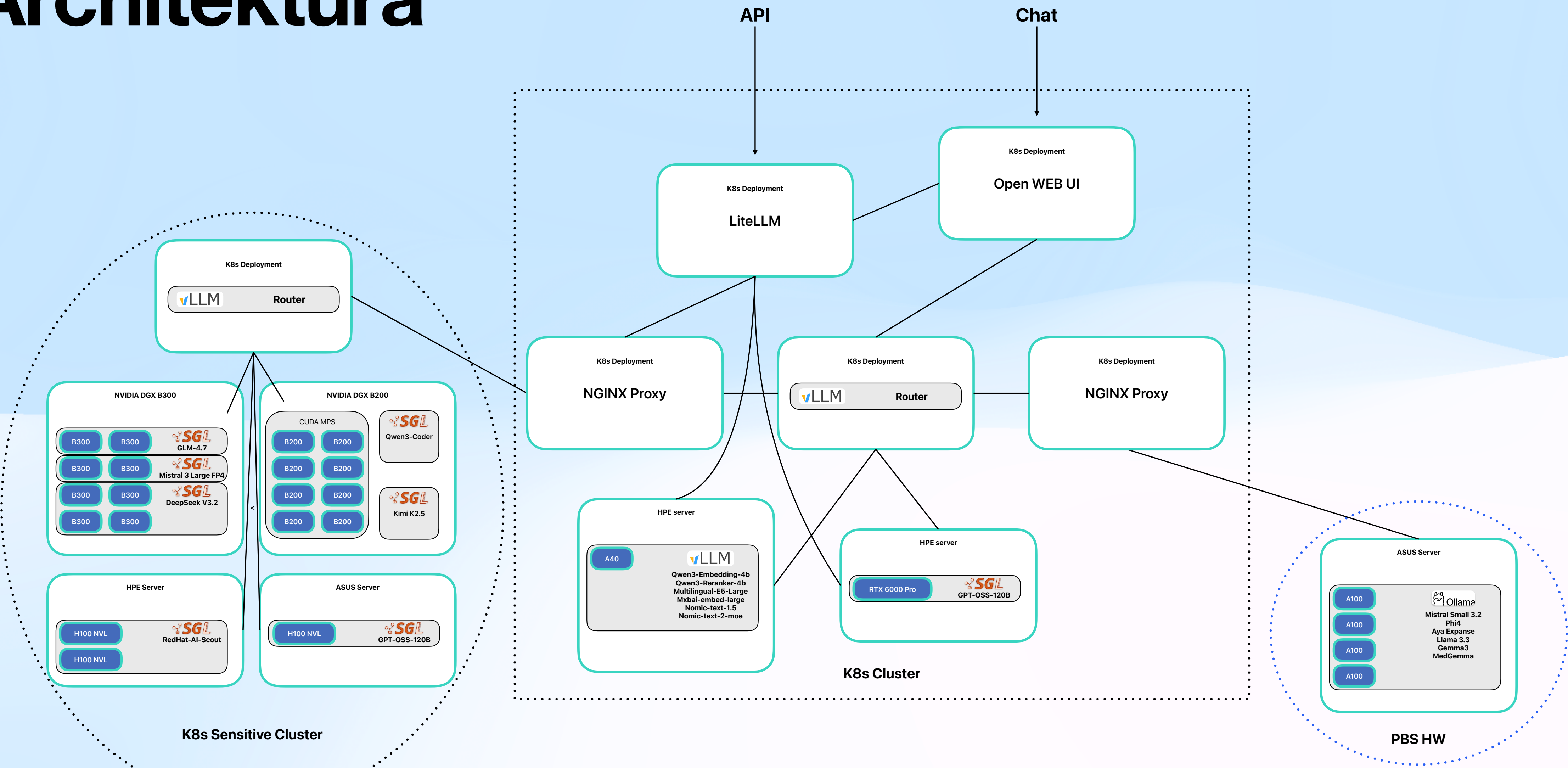
Porovnání modelů

Model	Max-total-tokens	GPU Mem	Tokens/s	Aider polyglot
Kimi-K2-Thinking	262144	788.8 GB	152	76.0 %
Kimi-K2.5	262144	793.8 GB	155	87.1 %
DeepSeek V3.2	163840	774.7 GB	86	80.5 %
Mistral 3 Large	163840	822.0 GB 532.0 GB (fp4)	80	46.7 %
GLM-4.7	163840	475.6 GB	68	68.4 %
GPT-OSS-120B	231168	92.5 GB	205	54.2 %

Řešené problémy

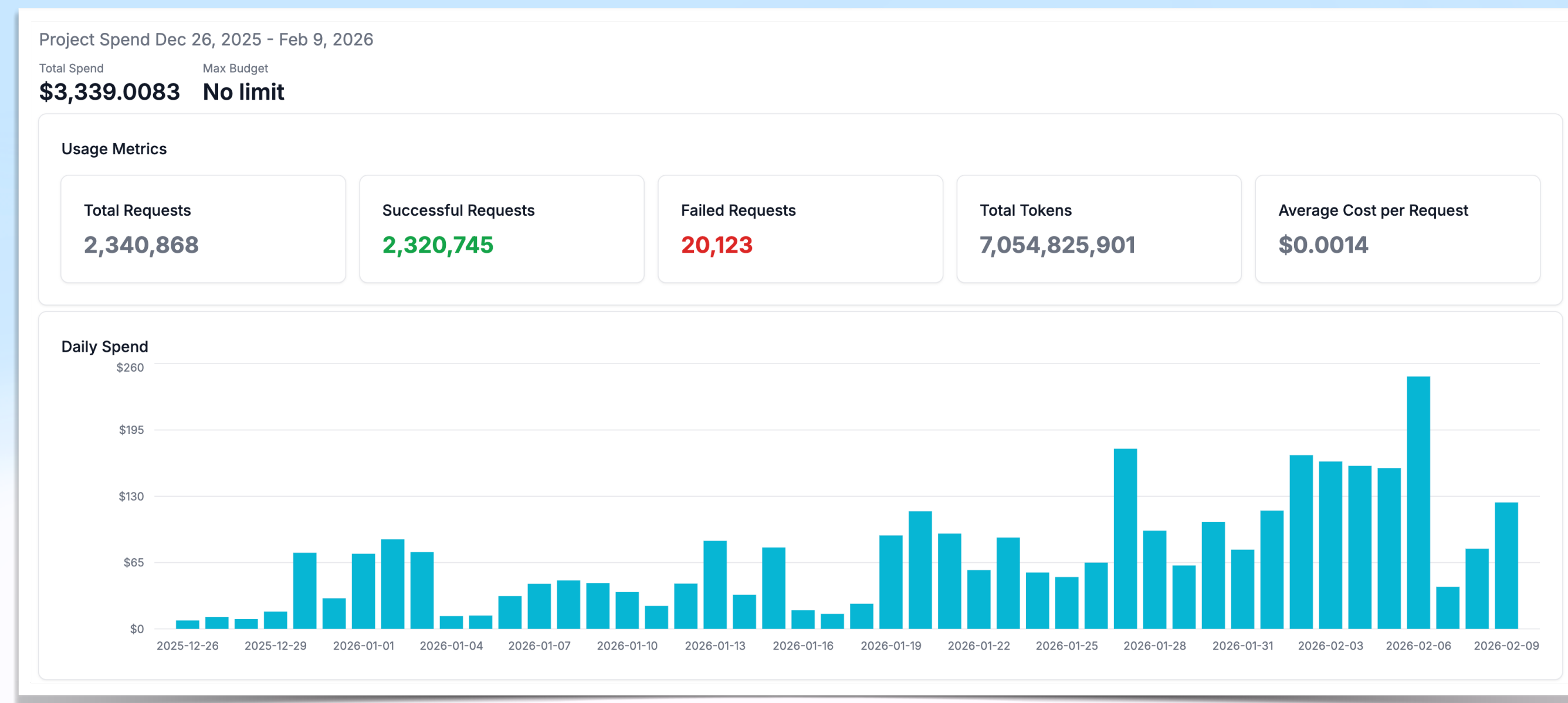
- Ollama
 - Uživatel může změnit velikost kontextu = restart modelu
 - Statická rezervace paměti pro paralelní požadavky
 - Nedostatečné metriky
- SGLang
 - Paralelní start instancí na jedné GPU selhává — nesmí se měnit volná paměť
 - Některé modely startují velmi dlouho (např. Kimi-K2.5)
 - RTX Pro 6000 nemá implementaci pro FP4
 - Některé FP8 modely nestabilní
 - Ne všechny modely škálují s počtem GPU
- vLLM router
 - Neposkytuje užitečné metriky

Architektura



LiteLLM

- Statistiky využití
 - Lepší přehled o uživatelích
 - Simuluje ceny
- Rate limit uživatelů
- Je schopno přepisovat požadavky
 - Vynutit (non)reasoning
- Kompatibilní API
 - Transformace Anthropic na OpenAI



Integrace do infrastruktury

- Jupyter Notebooky
- Vibe-coding aplikace
 - <https://deepsite.ai.e-infra.cz>
- Podpora agentů
 - Claude Code
 - Open Code
 - Codex
- MCP servery

Jupyter Notebook

The screenshot displays the Jupyter Notebook interface. The top menu bar includes File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. The left sidebar contains the Notebook Intelligence panel, which provides a summary of the current cell's content. The main area shows a code cell with the following Python code:

```
[3]: import pandas as pd

# Load the CSV file
df = pd.read_csv('age_data.csv')

# Display the first few rows
df.head()
```

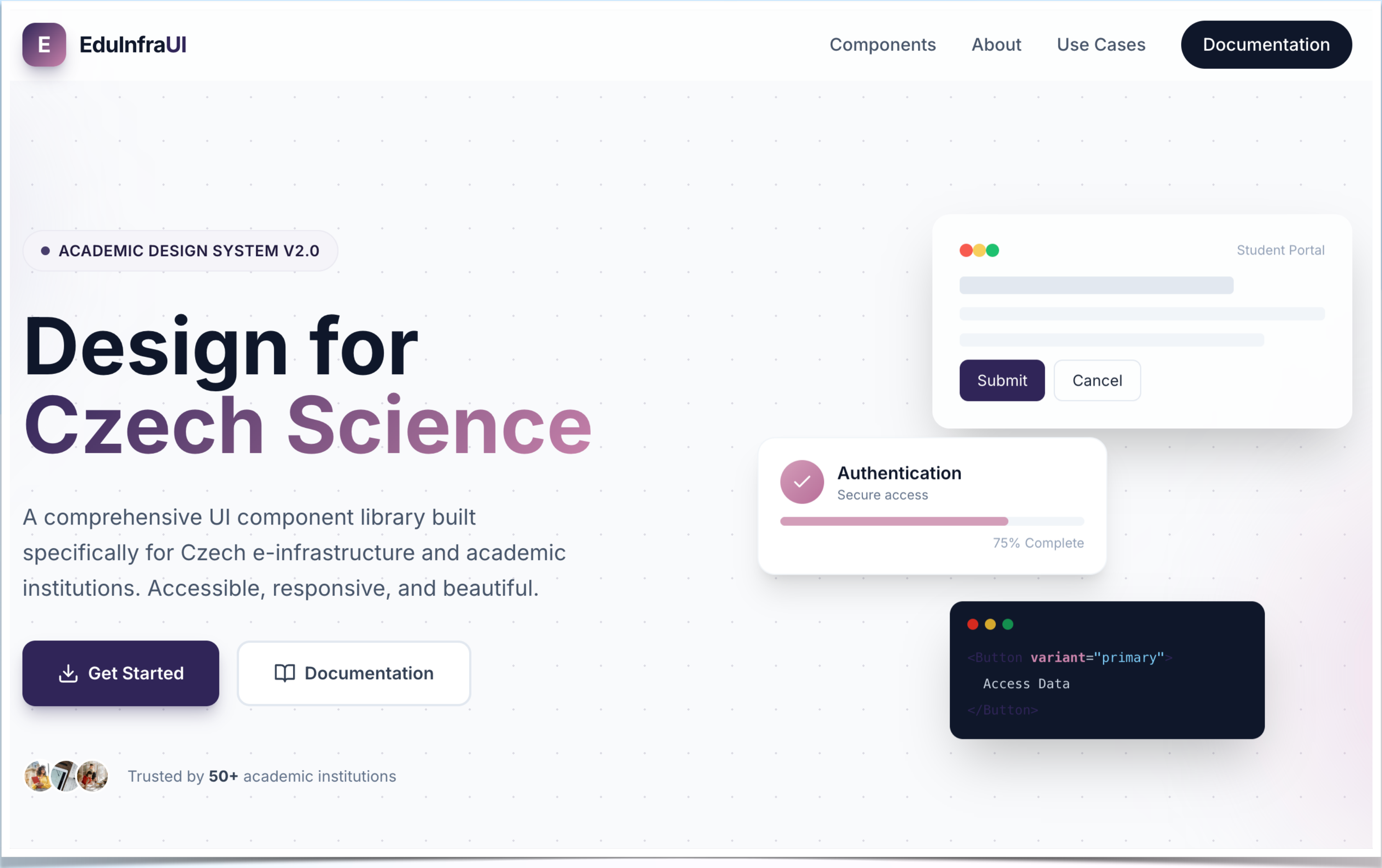
The output of the code is a table showing the first five rows of the CSV file:

	sex	age	city	country
0	Male	64.0	Houston	Brazil
1	Female	NaN	San Jose	Mexico
2	Male	68.0	San Antonio	Mexico
3	Male	72.0	Dallas	USA
4	Male	81.0	San Antonio	Argentina

The Notebook Intelligence panel on the left provides a detailed summary of the code and its output. It states: "(no content)" and "Perfect! I've successfully loaded your 'age_data.csv' file and displayed the first 5 rows. The data shows:" followed by a list of 4 columns (sex, age, city, and country) and a data sample. The data sample includes Row 0 (Male, age 64, Houston, Brazil), Row 1 (Female, age missing, San Jose, Mexico), Row 2 (Male, age 68, San Antonio, Mexico), and so on. It also notes that there are missing age values (NaN) and that the notebook has been saved.

The bottom status bar shows the notebook is in "Simple" mode, using the "Python [conda env:base] *" kernel. It also displays system metrics: CPU: 0.00 %, Mem: 809.44 / 8192.00 MB, and the current mode is "Command". The cursor is at line 1, column 1.

DeepSite



Agenti a vývojářské nástroje

- Claude Code
 - Aktuálně pravděpodobně nejpopulárnější nástroj
 - Nativně míří na Anthropic API a jeho modely
 - Lze přesměrovat na LiteLLM API a lokální modely
 - Kimi-K2.5 má nejlepší výsledky
 - Integrace do Jupyter, Rstudio, VS Code
- Open Code / Codex
 - Nativně používají OpenAI API
- Požadavky na modely
 - Správná podpora volání nástrojů (tools)

MCP servery — co jsou zač

- Dodávají modelům kontext
 - Dohledávání znalostí
 - Např. Web search / web fetch
- Mohou provádět akce
 - Spouštění lokálních programů
 - Volat vzdálených API

MCP servery — příklady

- DuckDuckGo Search
 - Vyhledávání a stahování webových stránek
- DocFork
 - Vyhledávání dokumentací na githubu
- K8s Cerit
 - Ukázky, validace a opravy K8s manifestů dle našich pravidel
- NPMJs
 - Prohledávání balíčků, verzí a dokumentace
- Prolog
 - Logické odvozování
- Shadcn / Tailwind
 - Přesné informace o UI komponentách
- Solver
 - Matematické úlohy, soustavy rovnic, statistika

MCP servery – implementace

- Část převzatá z hotových řešení na GitHubu
- Část vytvořena pomocí Claude Code
- Všechny MCP běží v K8s
- Napojeny na LiteLLM

Děkuji za pozornost

k8s@cerit-sc.cz

<https://blog.e-infra.cz>