# Prečo teda grafová databáza?

* Chceli sme sa oboznámiť s grafovámi databázami, lebo v určitých prípadoch použiťia vedia byť omnoho rýchlejšie, ako zaužívané relačníé databbázy
* v prípadoch kedy teba riešiť zložitejšie zvstahy v dátach
* Ako typické príkaldy použítia sa uvádzajú sociálne siete, logistické systémy ale aj osporúčacie systémy na produkty v internetových obchofdoch

Relačné databázy v tých to prípadoch zaostávajú z dvoch hlavných dôvodov,

* za prvé preto, lebo pri prehladávaní vstahu ktorý ide cez N tabuliek je treba vykonať N prehladaní tabuľky indexu, čím rýchlo narastá komplexita vyhladávania a systém sa spomaluje
* za druhé preto, lebo pri komplexnej štruktúre dát s monhými vstahmi je nutné pridávat kopu spájacích tabuliek a tým klesá prehladnosť dátového modelu a

# Výber grafovej databázy

* Pri výbere databáz sme zobrali troch najpopulárnejších predstaviťeľov grafovoých databáz
* NEO4j
  + POPULARITA
    - Z čoho vyplíva dobrá integrácia s najpoužívanejšímy frameworkami a jazykmi, dobrá online dokumentácia a pomohli vydaŤ aj odbornú literatúru
  + CYPHER
    - Dopytovací jazyk Cypher ktorý je Velmi intuitývny ,ktorý sa nesnaží nesilu držať SQL synaxe, no práveže využíva grafické prvky ako rôzne zátvorky na reprezentáciu vrcholov a šípky na reprezentáciu hrán, čo velmi zvyšuje prehľadnosť dopytov
* OrientDB
  + Rýchlosť
    - Podľa viacerých zdrojov je rýchlejšia ako Neo4J no v kontexte porovnania grafovej databázy s iným druhom toto nehralo až tak dôležitú rolu, lebo som vychádzal z predpokladu, že akákoľvek grafová databáza bude v našom prípade rýchlejšia ako relačná
    - Podporuje aj dalšie typy úložiska ako key-value store a dokumentovú databázu no to opať v našom kontexte nemalo využitie
    - Ako nevýhosu som videl použitie dopytovacieho jazyka SQL alebo veľmi podobného SQL, čím z môjho pohľadu znížila prehľadnosť dopytov
* Titan project
  + Táto databáza bola v zozname Dbrank ako tretia najpoužívanejšia, no je určená hlavne na distribuované systémy, čo v našom prípade nebolo vhodné lebo sme pracovali na jdnom servri
* !! vybrali sme teda Neo4J

# Aplikácia

* Aplikácia ,ktorú sme navrhli, a implementovali má slúžiť cestovatelovi ako pomôcka pri plánovaní trasy.
* Aplikácia umožnuje používateľovi sa zaregistrovať., príhlásitť., po prvom prihlásení si musí vybrat domáce mesto ktoré bude jeho východzie, dalej si vie vyhladaŤ a priávaŤ do zoznamu destinácie ktoré chce navštíviť. K rpidaným destináciam systém vyhladáa rôzne spôsoby dopravy ktoré sú vedú z domáceho miesta do destiáciea a

# Implementácia

* Aplikácia je postavena na jazyku PHP na frameworku Laravel, ktorý je postavený na vývojovom vzore Model View Controller.
  + Ako databázu sme použili komunitnú edíciu databázy Neo4J. Na komunikáciu medzi PHP frameworkom a databázou som použil najprv knižnicu NeoEloquent

Táto knižnica umožňuje integráciu grafovej databázy Neo4J do ekosystému Laravel a uľahčila vytváranie a implementáciu štruktúry databázy.

* + Táto knižnica však vedela naplno využiť rýchlosť Neo4J pri zložitejších dopytoch ako bolo napríklad. zobrazenie odporúčaní, ktoré si vyžadovalo složitejšie dopyty na databázu.
* Na tieto zložitejšie dopyty sme neskôr použili PHP Neo4J Client, lebo umožnuje prístup k natívnemu jazyku Neo4J CYPHER, ktorý vie naplno využiť rýchlosť databázy aj pri složitých dopytoch.
* Všetky údaje potrebné na funkčnosť aplikácie sme brali z API rome2rio. medzi tieto údaje patria:
  + GPS súradnice a názvy destinácií, v akých provinciách a štátoch sa rôzne destinácie nachádzajú.
  + toto api nám umožňovalo vyhľadanie trasy medzi jednotlivými destináciami a poskytlo informácie o trase medzi jednotlivými destináciami a ko rôzne typy dopravy, rôznych dopravcov a ceny všetkých spôsobov dopravy

# Výpočet TSP pomocou genetického algoritmu ako alternatíva k terajšiemu riešeniu?

Mal som geneticke algo na PHP urovni