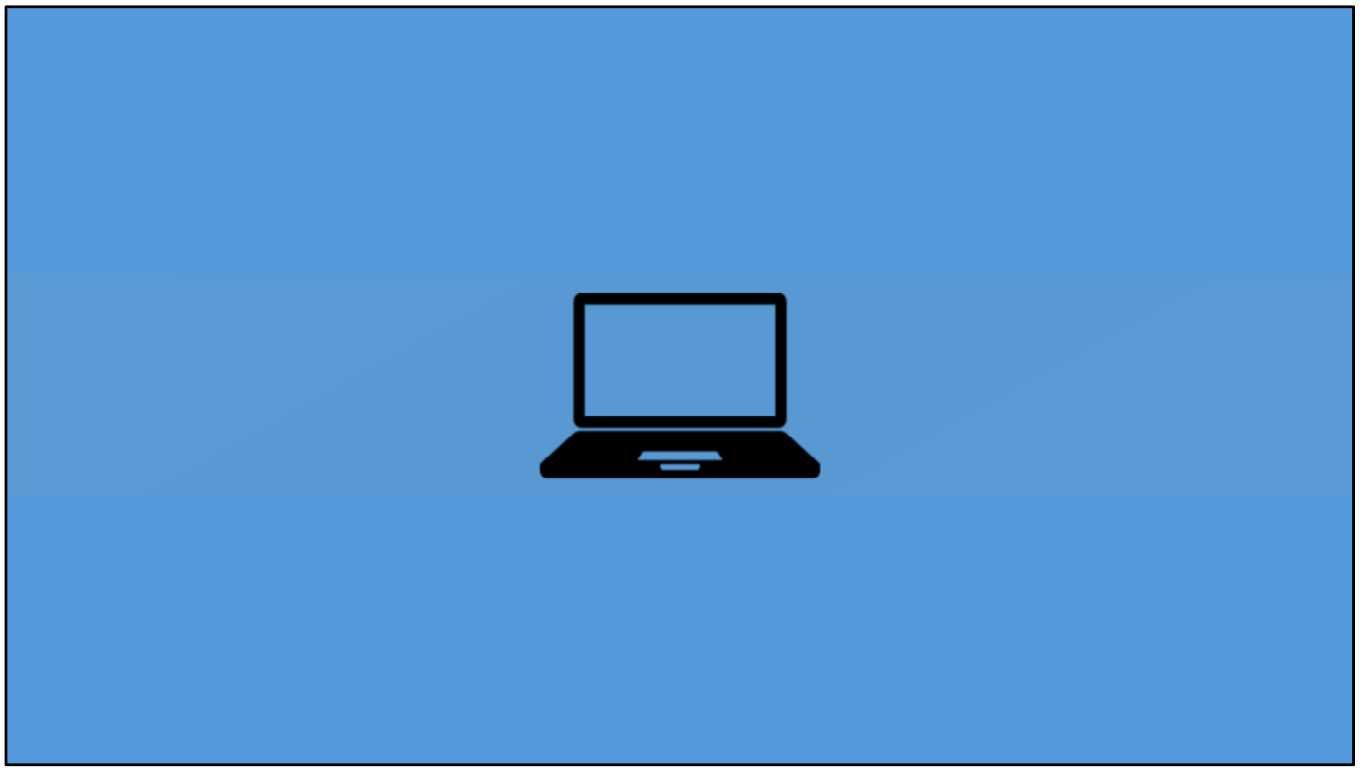
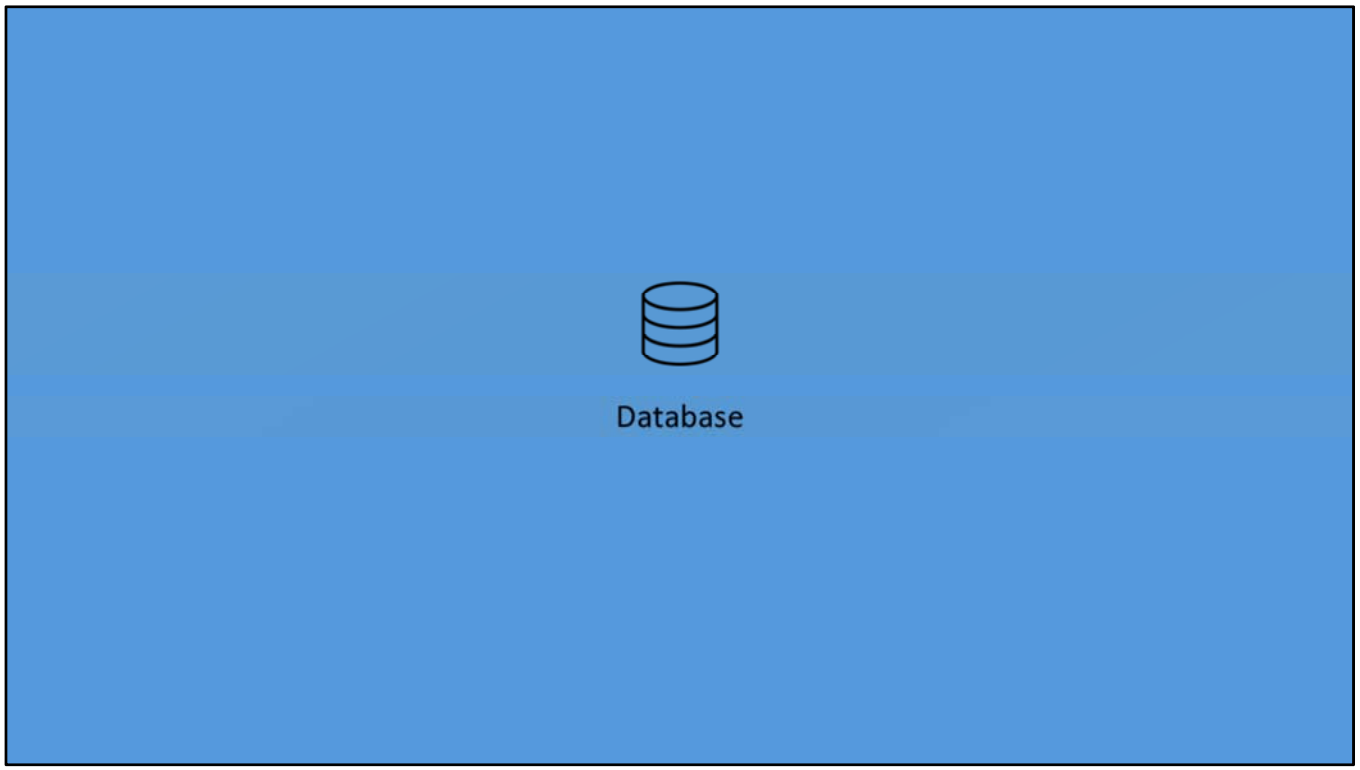
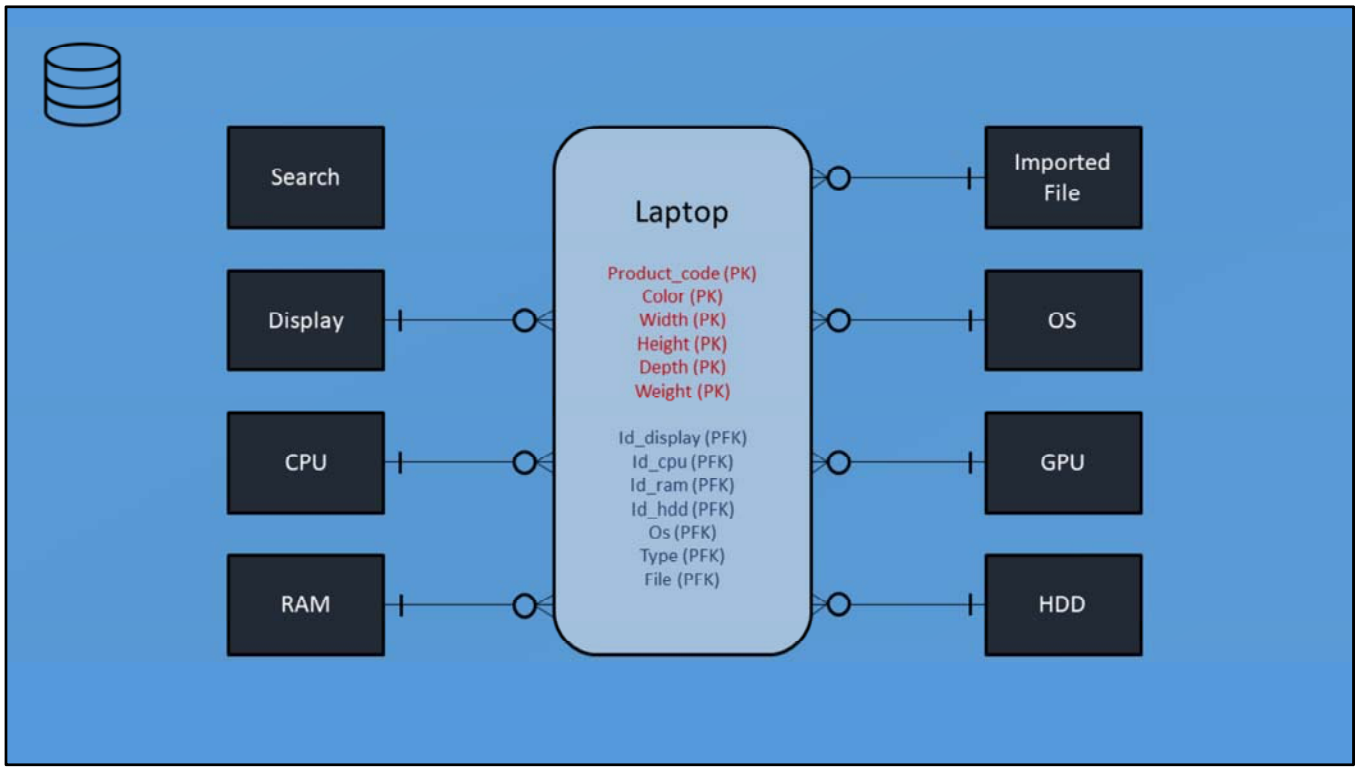


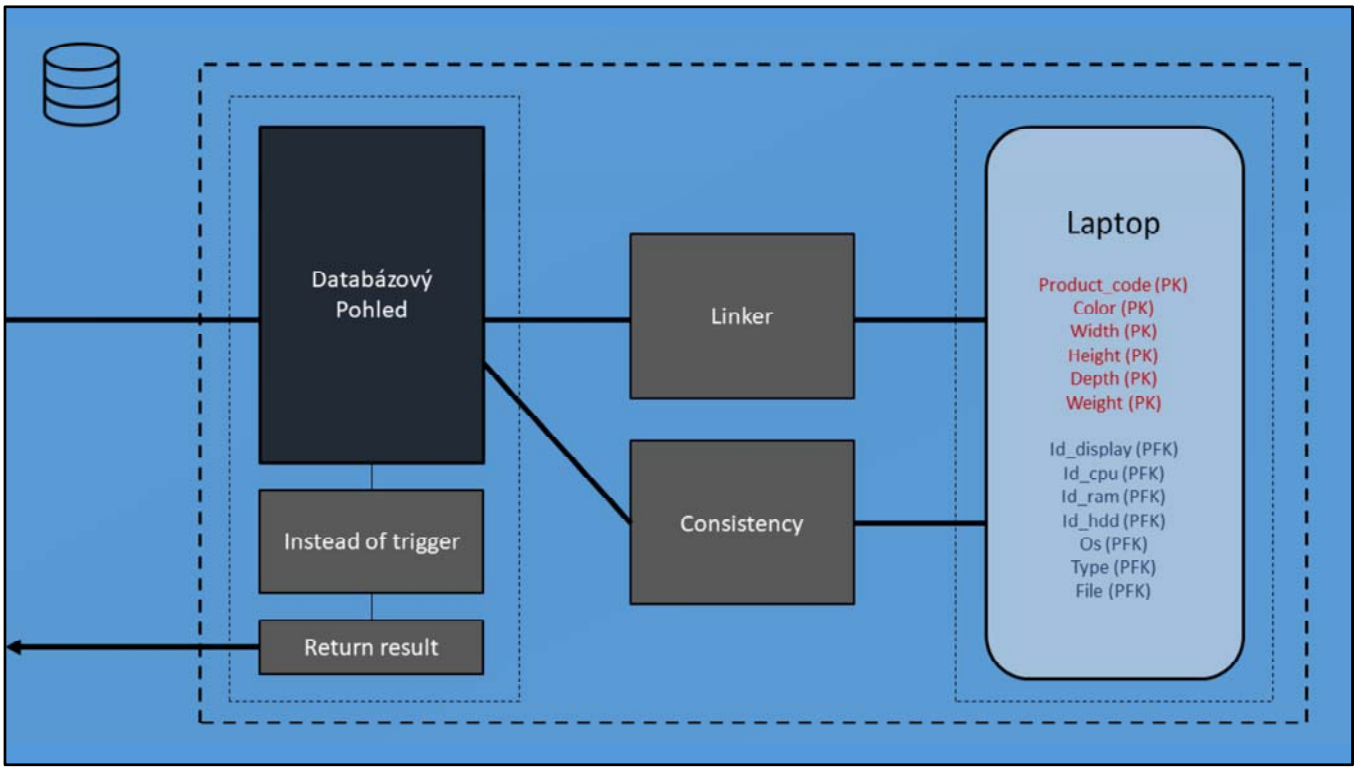
Dobrý den, nyní vám představím náš projekt na předmět řízení databází, který byl rozdělen do 3 fází.



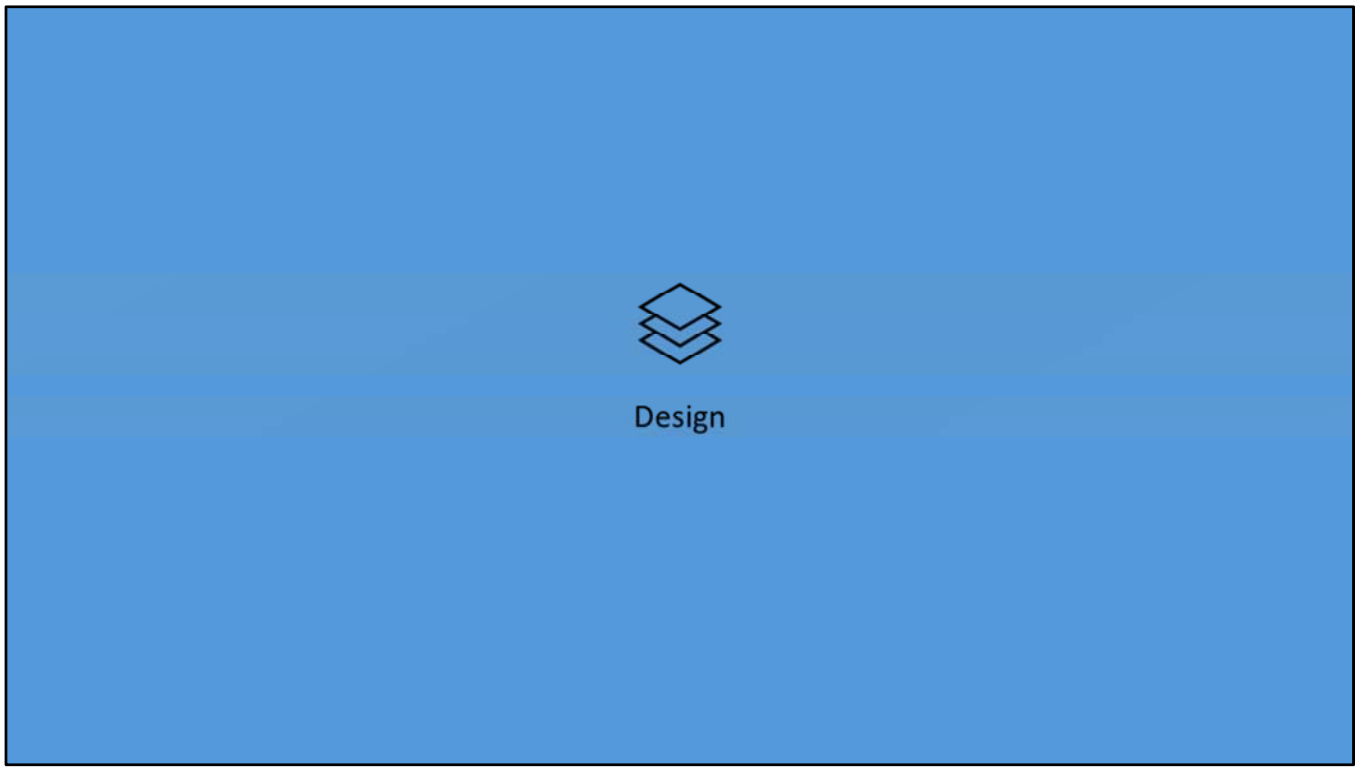
První fází byl návrh a implementace databáze.



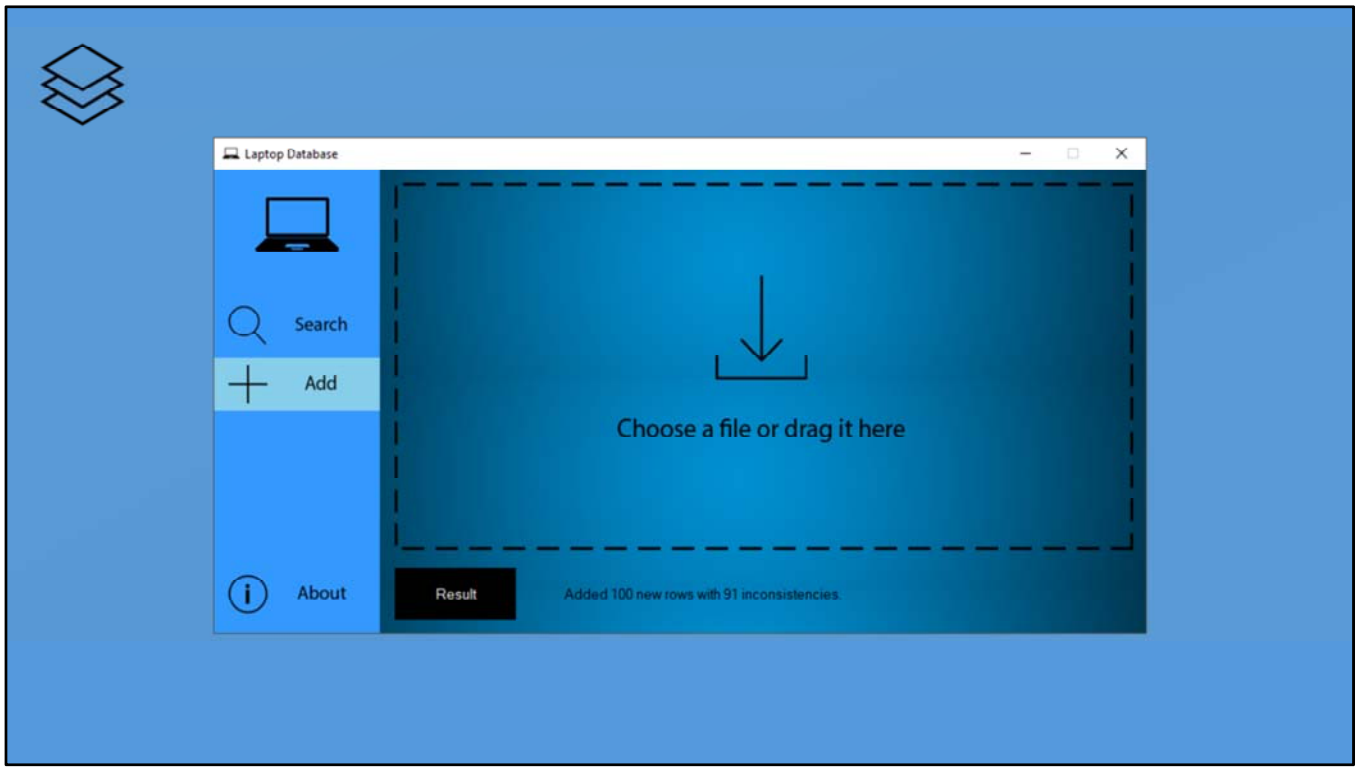
Databáze byla navrhnuta tak, že každá část notebooku má vlastní tabulku, která obsahuje jedinečné ID a všechny parametry, například RAM obsahuje typ, frekvenci a velikost. Všechny tyto tabulky se spojují do jediné velké M:N tabulky, která vše spojuje do jednoho celku. Hlavní tabulka navíc ještě kromě jednotlivých částí obsahuje rozměry jednotlivých modelů a unikátní produktové číslo. Nakonec je v databázi tabulka, obsahující veškerou historii vyhledávání a tabulka s hashy jednotlivých přidaných souborů.



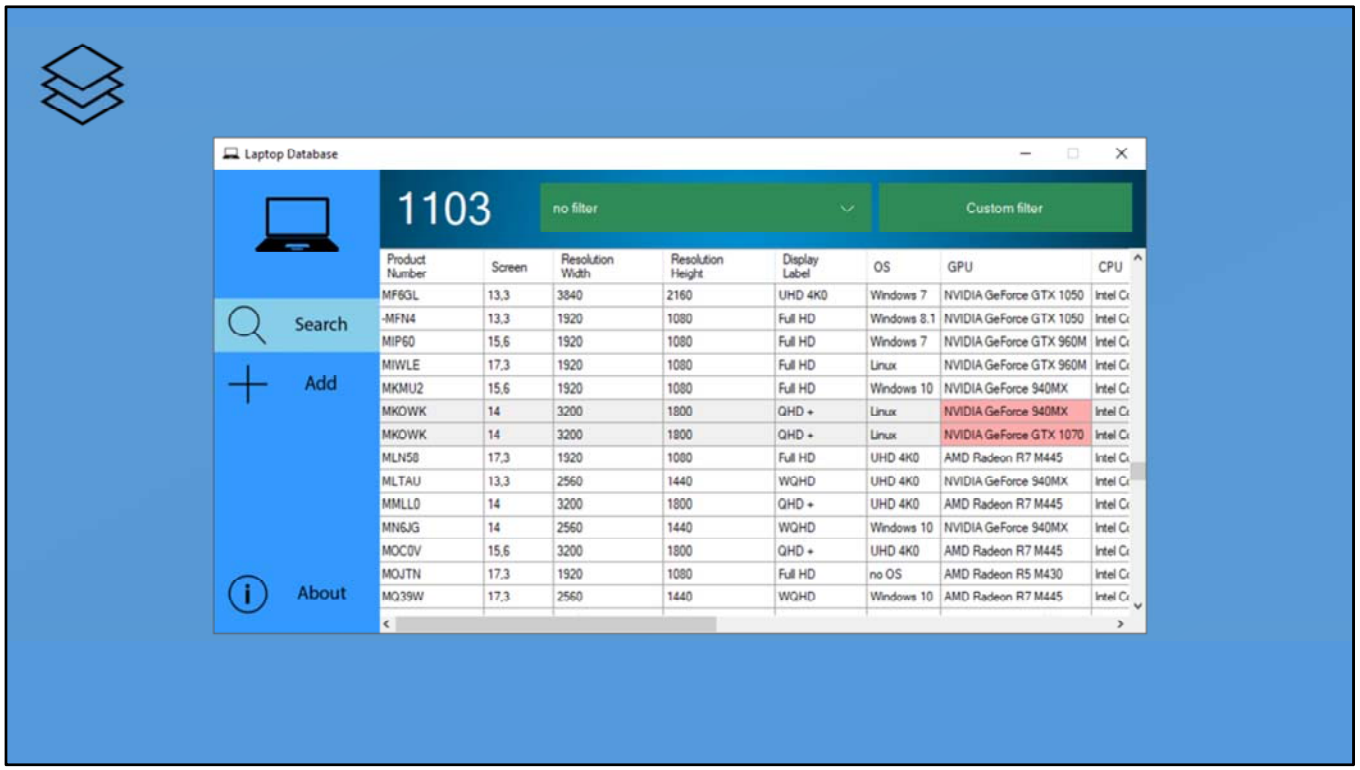
Práce s daty je řešena v databázi. Přístup do databáze zastřešuje pohled, který pracuje se dvěma funkcemi. První je Linker, který propojuje jednotlivé tabulky do jedné. Druhou je Consistency, která zjišťuje konzistence jednotlivých notebooků a to přes zjištění počtu unikátních prvků. Pokud je počet větší než 1, není konzistentní. Vkládání dat probíhá přes instead of trigger na pohledu, kterému se předají údaje a ten podle nich buď přidá nový notebook, nebo nepřidá v případě, pokud již notebook v databázi existuje. Pro navrácení výsledku importu byla vytvořena procedure Return result, který vrací výsledek pomocí print, a ten se zachycuje eventem v aplikaci.



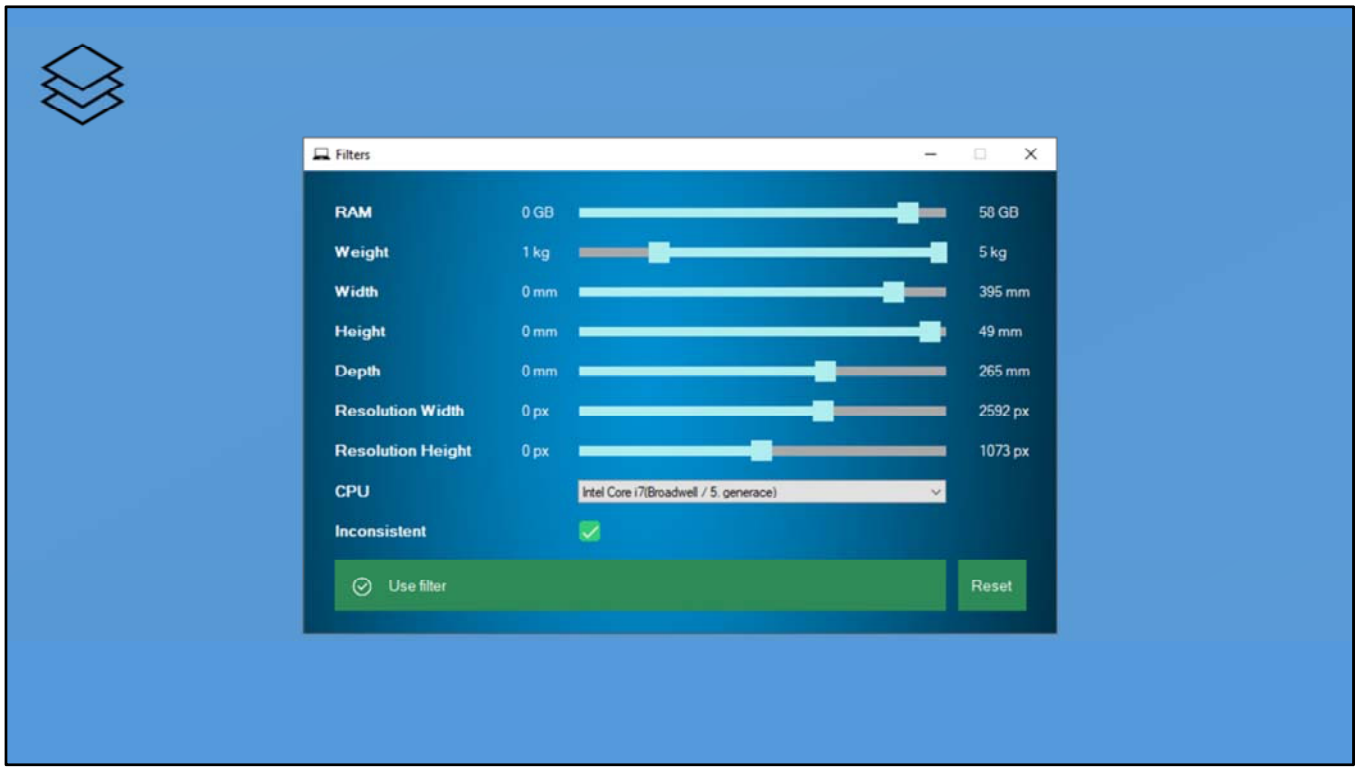
Druhou částí byla tvorba prostředí, tedy design.



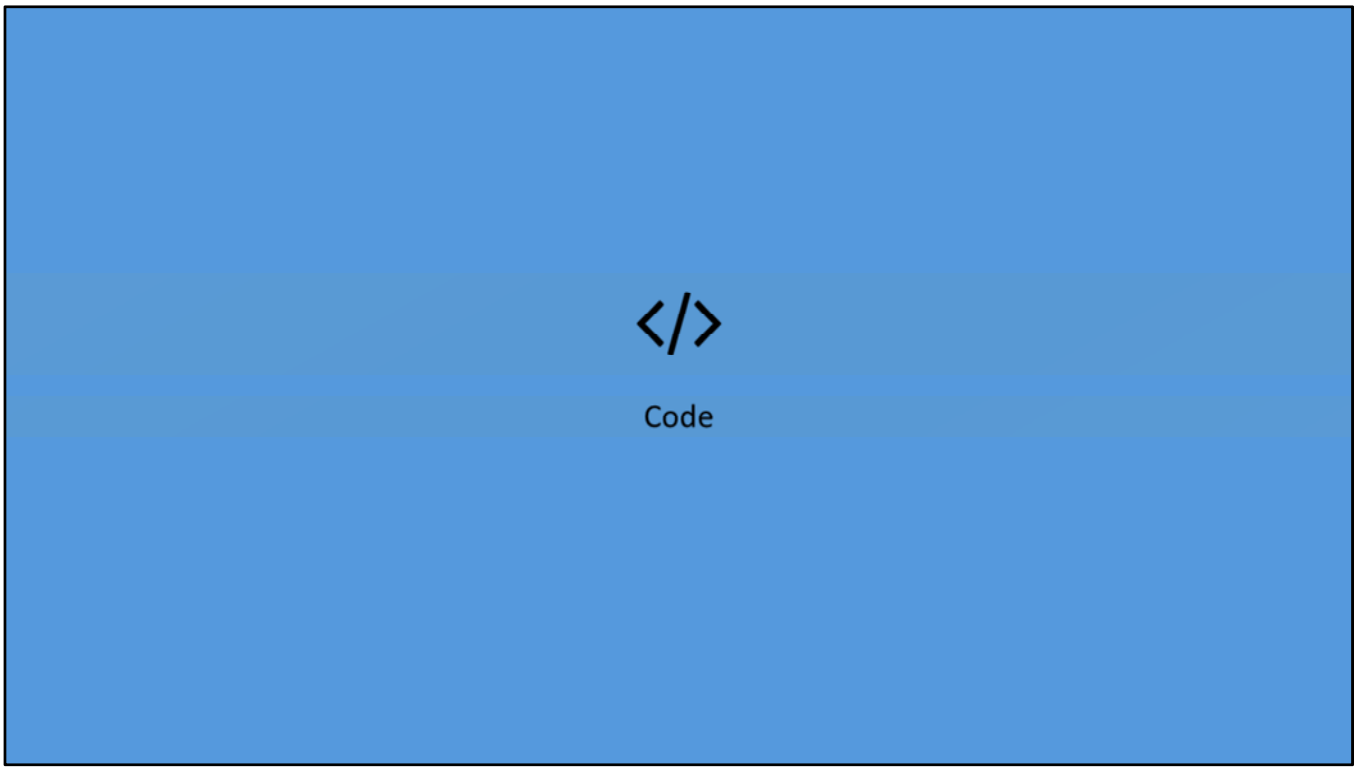
Pokud přeskočím jednotlivé animace a přechody, tak se dostanu k vkládání dat, která lze vkládat pomocí záložky Add. Vložení je možné jak kliknutím a vyhledáním souboru, tak pomocí DragAndDrop. Po zpracování souboru v jiném vláknu, se zobrazí počet nově přidaných a nekonzistentních řádků, nebo případně informace o tom, že soubor byl již jednou přidán a nelze ho vložit znovu.



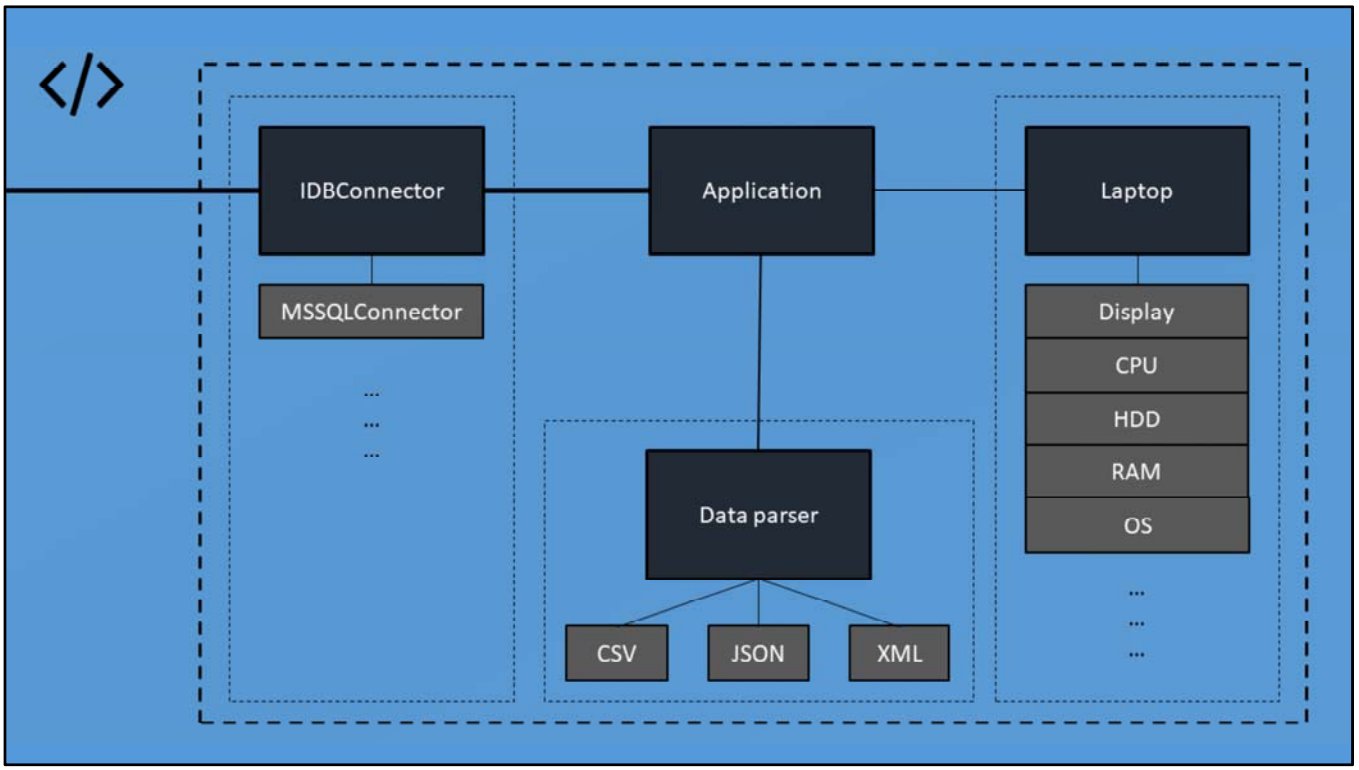
Hlavním oknem je vyhledávací okno, kde lze zobrazovat data z databáze, lze data filtrovat a lze graficky vidět nekonzistence. Vždy je vidět počet aktuálně zobrazených řádků a aktuálně zvolený filtr, který si lze navolit sám, nebo si vybrat z 10ti nejpoužívanějších. Každý nekonzistentní řádek je zvýrazněn šedivou barvou a každá nekonzistentní hodnota je zvýrazněna červeně. Jednotlivé řádky, které jsou nekonzistentní jsou vždy pod sebou.



Samotné filtrování je možné nastavit dle zadání, buďto jako rozsah hodnot, vybranou hodnotu, nebo maximální hodnotu. Zobrazené hodnoty jsou vždy aktuální s hodnotami v databázi, tedy maxima, minima i samotné hodnoty vždy odpovídají realitě.



Poslední, třetí částí, je kód samotné aplikace.

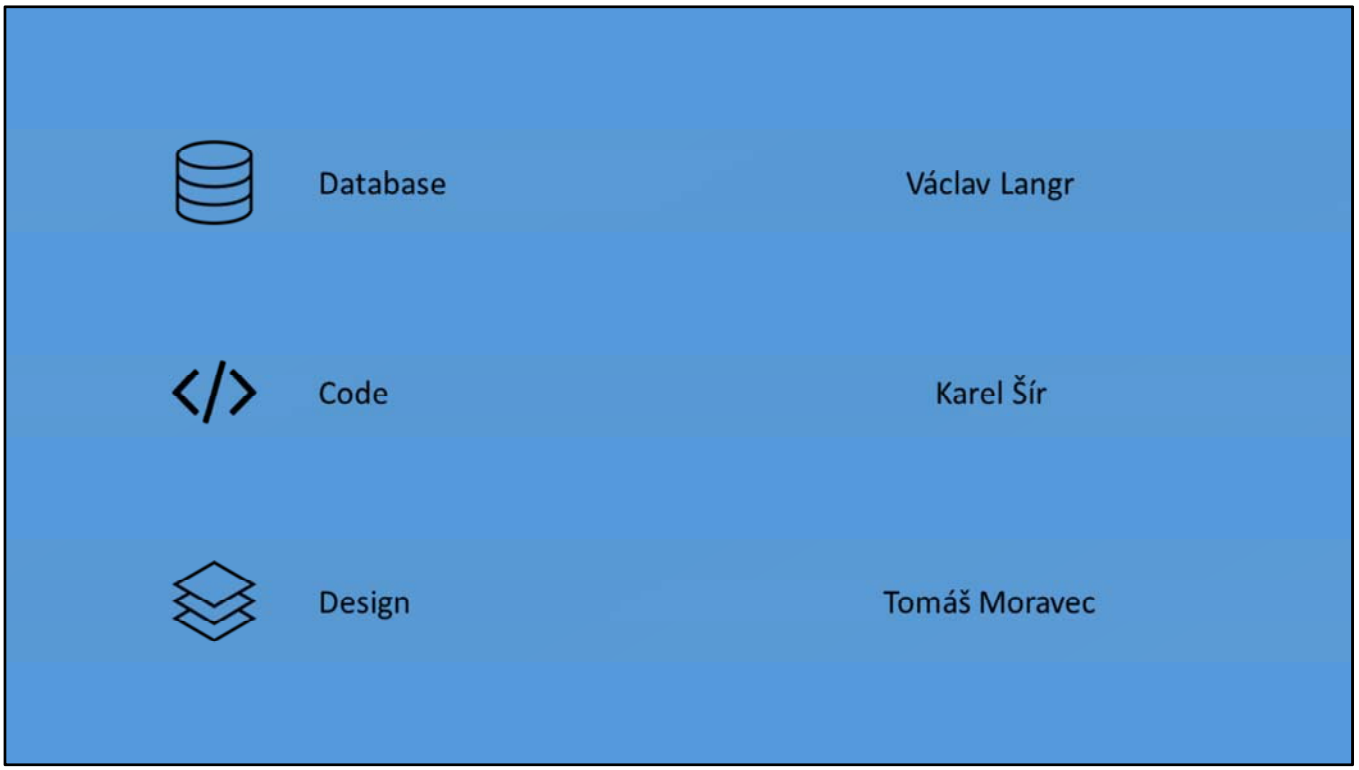


Na obrázku vidíte blokové rozvržení, které odpovídá třídám aplikace. Pokud začnu zleva, data jsou z databáze načítána pomocí Konektoru, který staví na interface, díky čemuž lze napojit i jiné typy databází. Konektor poskytuje veškeré třídy nutné k importu a čtení z databáze, včetně získávání podrobnějších informací, jakou jsou hodnoty pro filtr, protože veškerá data se získávání z databáze, nikoliv z načtených hodnot v aplikaci.

Dále samotná aplikace implementuje veškerou komunikaci s konektorem, tedy import, zobrazení dat do tabulky, zvýraznění nekonzistencí a hlavně o filtrování dat.

Při importu se využívá Data Parser, který umí zpracovat 3 typy vstupních souborů. CSV, JSON a XML.

Samotná data jsou načítána do objektů, které jsou identické s databázovým modelem.



Každý v našem teamu měl své primární zaměření, nicméně o každém problému jsme společně diskutovali a dosáhli tak nejlepších možných řešení.
Děkuji za pozornost.