FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Skladový informační systém

Návrh architektury a databázový model

Dokument vytvořen pro potřeby předmětu BI-SI1

Autoři: Róbert Selvek, Vojtěch Cahlík, Josef Hušek, Jan Lidák

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Obsah

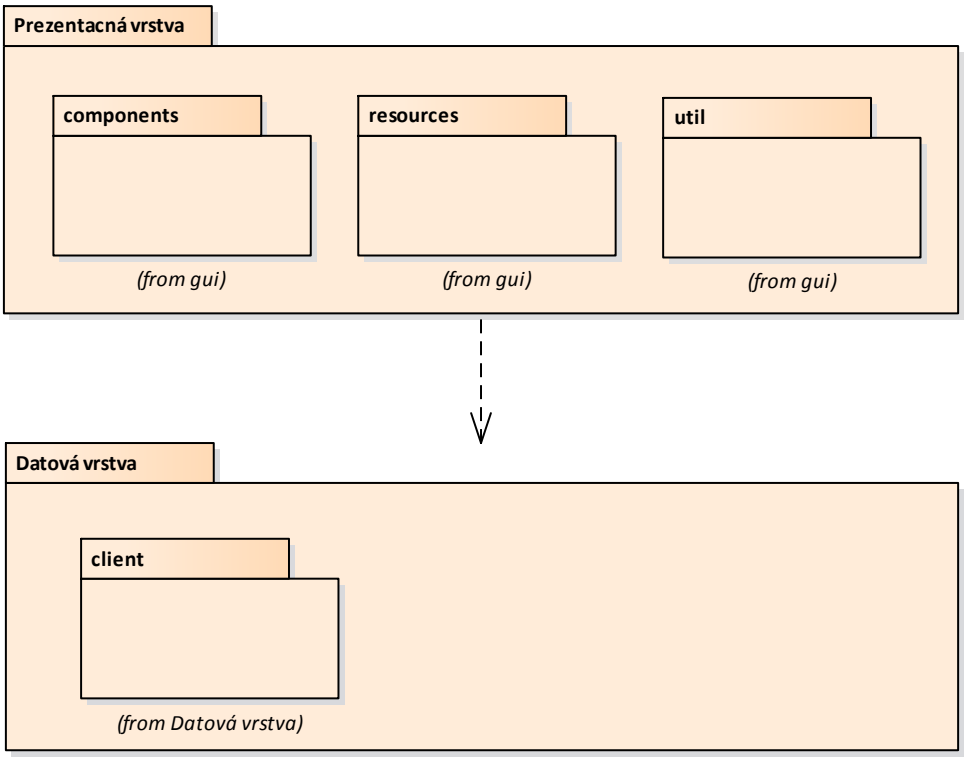
1. Návrh logickej architektúry 3
1.1 Klientská cast 3
1.1.1 Datová vrstva 3
1.1.1.1 client 4
1.1.2 Prezentacná vrstva 4
1.1.2.1 gui 4
1.1.2.1.1 components 4
1.1.2.1.2 resources 4
1.1.2.1.3 util 5
1.2 Serverová cast 5
1.2.1 Business Layer 6
1.2.1.1 operations 6
1.2.2 Data Layer 6
1.2.2.1 core 7
1.2.2.2 db 7
1.2.3 Presentation Layer 7
1.2.3.1 auth 7
1.2.3.2 representations 7
1.2.3.3 resources 8
2. Relační datový model 9
2.1 line_items *«table»* 10
2.2 order_ins *«table»* 10
2.3 order_outs *«table»* 10
2.4 orders *«table»* 10
2.5 product_movements *«table»* 10
2.6 products *«table»* 11
2.7 users *«table»* 11

1. Návrh logickej architektúry

Informacní systém SKLD se skládá ze dvou oddelených cástí - serveru a klienta. Tyto dve cásti spolu komunikují pres REST API a jsou spravovány jako oddelené projekty.

1.1 Klientská cast

Klientská cast systému je tvorená dvojvrstvou desktopovou aplikací, napsanou v jazyce Java. Skládá se z prezentacní a datové vrstvy, prezentacní vrstva zajištuje zobrazení a logiku pruchodu uživatelským rozhraním, zatímco datová vrstva zajištuje komunikaci se serverem a zpracování dat. Datova vrstva je umistena v samosatatné knihovne, kterou projekt implementujici prezentacni vrstvu importuje. Obe vrstvy jsou oddeleny rozhraním, které vystavuje datová vrstva. Komunikace v rámci tohoto rozhraní používá trídy z balícku representations ze serverové cásti systému. Závislost na tomto balícku není v diagramu logické architektury klientské cásti vyznacena, vyplývá však z diagramu logické architektury serveru.



Obrázek 1 - Logická architektúra klientskej casti

1.1.1 Datová vrstva

Datová vrstva aplikace vystavuje prezentacni vrstve rozhrani pro komunikaci se serverem. Datová vrstva prezentacní vrstve bud ze serveru poskytne požadovaná data, nebo naopak na server odešle nejaký požadavek. Samotná komunikace se serverem probíhá pres REST API.

Datová vrstva je organizována jako samostatný projekt se samostatným build procesem atd. za úcelem prehledného oddelení odlišných knihoven používaných cástmi client a gui.

Použité frameworky a technologie:

- Sítová komunikace: httpclient, Jackson
- Správa buildu: Maven
- Testování: JUnit

1.1.1.1 client

Balícek client prebírá elementární požadavky z prezentacní vrstvy klienta, zabaluje príslušná data do formátu JSON a odesílá je prezentacní vrstve serverové aplikace pomocí REST API.

1.1.2 Prezentacná vrstva

Prezentacní vrstva aplikace je zodpovedná za správu a logiku uživatelského rozhraní a jeho zobrazování uživateli. Tomu umožnuje na server odesílat požadavky, a naopak zobrazovat výsledná data. Komunikace vždy probíhá pouze pres rozhraní vystavené datovou vrstvou.

Prezentacní vrstva je tvorena samostatným projektem a obsahuje hlavní (tedy spustitelnou) cást klientské aplikace. Za úcelem výsledného sestavení spustitelné aplikace má nastavenu závislost na knihovnu datové vrstvy.

Použité frameworky a technologie:

- Uživatelské rozhraní (GUI): JavaFX
- Správa buildu: Maven

1.1.2.1 gui

Balícek gui obsahuje implementaci aplikace pro desktopové operacní systémy napsanou s použitím grafického frameworku JavaFX. Tato aplikace prezentuje uživateli data z datové vrstvy v podobe obrazovek s údaji, a naopak odesílá uživatelem zadaná data a príkazy na datovou vrstvu.

Gui obsahuje vlastními silami vytvorenou implementaci jednoduchého frameworku postaveného nad technologií JavaFX. Tento vlastní framework umožnuje pokrocilé procházení obrazovkami aplikace a znovupoužívání komponent na více místech. Koren balícku gui obsahuje základ tohoto frameworku, tedy obvykle abstraktní trídy znázornující grafické komponenty, které jsou pak používané skutecnou výslednou strukturou aplikace reprezentovanou balíckem components.

1.1.2.1.1 components

Balícek components obsahuje grafické komponenty používané v aplikaci, spolu s veškerou logikou a procházením GUI. Komponentou muže být napr. okno, obrazovka, ci jen jednoduchá cást okna nebo napr. formulár. Každá takováto komponenta je zde umístena v samostatném balícku a obvykle je poté v aplikaci znovupoužitelná na více místech.

Každá komponenta se skládá ze dvou cástí - hlavní cást (bez zvláštního pojmenování) se stará o veškeré logické procesy, komunikaci s ostatními komponentami aplikace, atd. Tato hlavní cást poté má vlastní "handler", který spravuje vlastní zobrazování grafických prvku technologie JavaFX.

1.1.2.1.2 resources

Balícek obsahuje trídy s globálními informacemi za úcelem zprehlednení kódu. Jedná se o cesty k XML souborum komponent JavaFX, nekteré textové retezce, konfiguraci, atd.

1.1.2.1.3 util

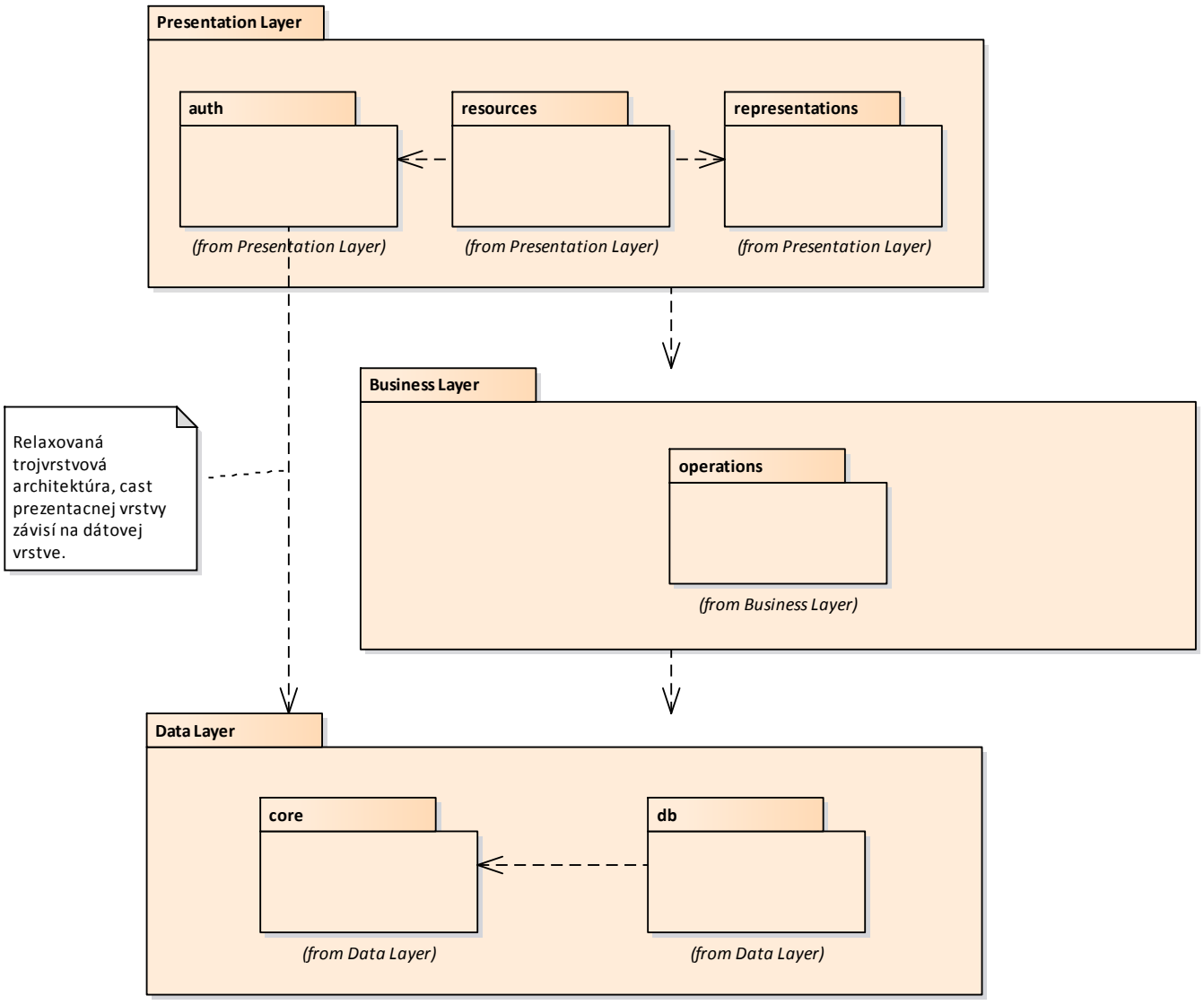
Balícek obsahuje takové pomocné metody a trídy, které se nehodí umístit obvyklým zpusobem, tedy k príslušné tríde. Zejména obsahuje funkce zjednodušující práci s technologií JavaFX.

1.2 Serverová cast

Serverová cast systému je implementovaná ako trojvrstvová aplikácia, s dátovou vrstvou abstrahujúcou prístupy ku databáze a konkrétne SQL dotazy, business vrstvou, ktorá implementuje zložitejšiu logiku systému a prezentačnou vrstvou, ktorá má na starosti routovanie HTTP požiadaviek, ich spracovanie a generovanie JSON odpovedí.

Knižnice použité pri tvorbe serverovej časti:a

- REST rozhranie: Dropwizard Framework
- Objektovo-relacné mapovanie: Hibernate Framework
- Zostavovanie a manažment závislost: Maven
- Testovanie: JUnit



Obrázek 2 - Logická architektúra serveru

1.2.1 Business Layer

Doménová vrstva je zodpovedná za business logiku aplikácie. Používa rozhrania DAO objektov na priamu úpravu dát a overuje pokrocilejšie business pravidlá (napr. "pri naskladnovaní objednávky nie je možné naskladnit viac kusov produktu, než bolo objednaných").

1.2.1.1 operations

Balícek operations implementuje triedy pre manipuláciu s hlavnými konceptami, s ktorými sa v aplikácií dá manipulovat - objednávky a produkty.

1.2.2 Data Layer

Dátová vrstva aplikácie zahrna triedy, ktoré umožnujú interakciu s databázou, v ktorej sa ukladajú dáta informacného systému. Na to obsahuje triedy reprezentujúce objektovo jednotlivé tabulky v databáze DAO objekty, ktoré umožnujú do databáze pristupovat.

Do databáze aplikácia nepristupuje priamo. Využíva štandard Java Persistence API a framework Hibernate na prevedenie objektovo-relacného mapovania a zabezpecenie prakticky úplnej databázovej nezávislosti. Nasadenie aplikácie však ráta s použitím databáze PostgreSQL.

1.2.2.1 core

Balícek obsahujúci definície entít, s ktorými aplikácia pracuje. Do znacnej miery kopíruje doménový model aplikácie.

Triedy reprezentujúce entity sú anotované podla štandardu JSR 338/JPA 2.1, co umožnuje použitie Hibernate Frameworku na tvorbu dotazov a konverziu z databázových záznamov na doménové objekty.

Názov core pochádza z konvencií REST Frameworku Dropwizard.

1.2.2.2 db

Balícek db obsahuje definície DAO tried. DAO triedy obalujú rôzne dotazy na databázu realizované pomocou frameworku Hibernate.

Názov db nasleduje konvencie frameworku Dropwizard.

1.2.3 Presentation Layer

Prezentacná vrstva definuje a implementuje REST rozhranie, pomocou ktorého frontend aplikácie implementované na rôznych platformách (webový klient, desktopový klient, mobilná aplikácia) dokážu komunikovat s informacným systémom a autentikovat užívatela.

1.2.3.1 auth

Balícek auth implementuje autentikáciu (prihlasovanie) užívatelov a ich autorizáciu (politiku prístupových práv). Na to implementuje frameworkové rozhrania io.dropwizard.auth.Authenticator a io.dropwizard.auth.Authorizer.

Autentikácia (v zmysle overovania tokenov, ktoré sa používajú na zistenie informácií o prihlásení užívatela) nie je súcastou business vrstvy, avšak priamo používa DAOs, aby získavala informácie o užívateloch.

Technicky je autentikácia užívatela implementovaná pomocou generovania a overovania JSON Web Tokens, predávaných v HTTP hlavickách požiadavok odoslaných na server.

Metódy na HTTP Resources používajú štandardné javax.annotation.security anotácie, ktoré indikujú frameworku, že musí overit, ci prihlásený úžívatel má dostatocné práva na prihlásenie.

1.2.3.2 representations

Balícek representations obsahuje triedy, ktoré sa serializujú z a do formátu JSON a tak slúžia ako požiadavky a odpovede na REST API, spracúvané balíckami resources a client.

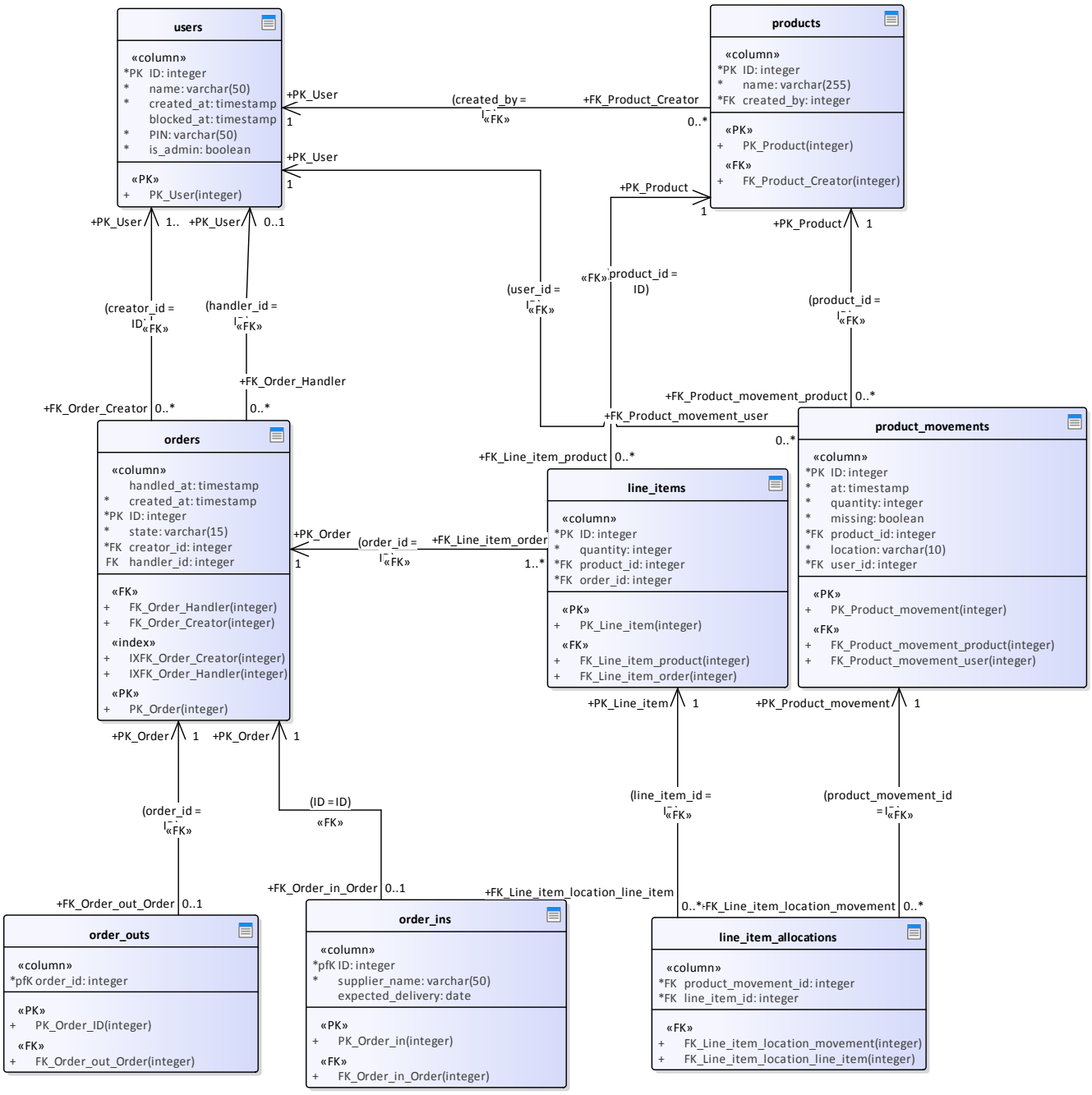
Tieto triedy neobsahujú takmer žiadnu funkcionalitu. Sú to len nositelia údajov a ako také sú zdielané medzi serverom a klientskou aplikáciou.

1.2.3.3 resources

Balícek resources obsahuje triedy, ktoré pomocou anotácií definovaných v štandarde JAX-RS a implementovaných knižnicou Jersey tvoria REST API knižnice.

Metódy v triedach zodpovedajú jednotlivým API endpointom a ich zodpovednostou je overovanie platnosti požiadavkov, získanie prislúchajúcich doménových objektov pomocou DAO tried a ich konverzia na reprezentácie.

2. Relační datový model



Obrázek 3 - Tables

2.1 line_items *«table»*

Predstavuje množství objednaných kusu jednotlivých produktu.

Název atributu	Datový typ	Not null	Popis
ID	integer	True	
quantity	integer	True	Pocet objednaných kusu produktu.
product_id	integer	True	
order_id	integer	True	

2.2 order_ins *«table»*

Objednávka produktu od dodavatelu e-shopu na sklad.

Název atributu	Datový typ	Not null	Popis
ID	integer	True	
supplier_name	varchar(50)	True	Název dodavatele nebo prepravní spolecnosti.
expected_delivery	date	False	Datum a cas ocekávaného prijetí objednávky na sklad.

2.3 order_outs *«table»*

Objednávka, která se bude vyskladnovat a expedovat zákazníkovi.

Název atributu	Datový typ	Not null	Popis
order_id	integer	True	

2.4 orders *«table»*

Požadavek na dorucení jistých produktu (se specifikovaným množstvím).

Název atributu	Datový typ	Not null	Popis
handled_at	timestamp	False	Datum a cas, kdy byla objednávka uzavrena.
created_at	timestamp	True	Datum a cas zadání objednávky.
ID	integer	True	
state	varchar(15)	True	Stav objednávky - otevrena, uzavrena, odmítnuta
creator_id	integer	True	ID uživatele, který objednávku vytvořil
handler_id	integer	False	ID uživatele, který objednávku uzavřel

2.5 product_movements *«table»*

Zaznamenává naskladnení nebo vyskladnení urcitého poctu kusu produktu na nejakém skladovém míste. Umožnuje také oznacení urcitého poctu kusu produktu jako chybejících.

Neoznacuje aktuální pocet kusu, místo toho funguje jako log presunu produktu.

Název atributu	Datový typ	Not null	Popis
ID	integer	True	
at	timestamp	True	Datum a cas kdy byl tento presun vykonán.
quantity	integer	True	Pocet kusu produktu, které byly na dané skladové místo pridány nebo z neho byly odebrány.
missing	boolean	True	Tyto produkty nebyly nalezeny na sklade. Nezobrazovat je v souctech produktu nachazejících se na skladovém míste.
product_id	integer	True	
location	varchar(10)	True	
user_id	integer	True	ID uživatele, který produkty přesunul

2.6 products *«table»*

Jeden typ produktu, který se dá uložit do skladu.

Název atributu	Datový typ	Not null	Popis
ID	integer	True	
name	varchar(255)	True	Název, který produktu dal dodavatel.
created_by	integer	True	ID uživatele, který produkt v systému vytvořil.

2.7 users *«table»*

Reprezentuje uživatele v systému

Název atributu	Datový typ	Not null	Popis
ID	integer	True	
name	varchar(50)	True	Jméno uživatele
created_at	timestamp	True	Datum, kdy byl uživatel pridán do systému.
blocked_at	timestamp	False	Datum, od kterého už uživatel nesmí do systému pristupovat (protože s ním například byl ukoncen pracovní pomer)
PIN	varchar(50)	True	PIN, který uživatel použije na identifikaci a prihlášení se do systému
is_admin	boolean	True	Typ uživatele - Skladník nebo vedoucí smeny. Vedoucí má v systému plné práva zatímco skladník jenom omezené.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE