Body	Popis
100 spolu **+ 30 BONUS**	**Spring Boot MVC + Thymeleaf: Pizzeria** Vašou úlohou je vytvoriť webovú Spring Boot MVC aplikáciu s Thymeleaf na tému **Pizzéria**. Aplikácia bude obsahovať zabezpečené používateľské rozhranie s prihlásením a rôznymi prístupovými právami na základe definovaných používateľských rolí (zákazník, kuchár/manažér, kuriér, administrátor/majiteľ). Cieľom je **navrhnúť a implementovať systém**, ktorý umožní používateľom pracovať s údajmi vo viacerých databázových tabuľkách (napríklad správa pízz, ingrediencií, objednávok, používateľov, atď.) a zabezpečiť správnu interakciu medzi databázou a aplikáciou. Systém má pokrývať základnú funkcionalitu pizzerie: prehliadanie ponuky, vytváranie objednávok, spracovanie objednávok a správu používateľov a produktov. **Aplikácia MUSÍ byť v slovenskom jazyku, jedná sa o systém pre slovenský podnik.** Po dlhej úvahe som sa rozhodol, že projekt budete vypracovávať ako jednotlivci, nechcem riskovať Fx kvôli nesprávnemu rozdeleniu úloh.
	Funkčné požiadavky Systém musí umožniť: - prehliadanie ponuky produktov aj neprihláseným používateľom, - možnosť vložiť produkty do košíka (prihláseným používateľom, tzn. košík - serverovo závislý), - preklopenie košíka do objednávky prihláseným používateľom (zákazníkom), - správu objednávok zamestnancami podľa ich role (kuchár prijme odjednávku, vyrobí výrobok, pošle kuriérovi, ten prijme rozvoz, doručí, atď.), - pridávanie, úpravu a mazanie údajov oprávnenými používateľmi (pizza, ingrediencia, tag, konkrétna veľkosť pizze, atď.), - správu používateľov, rolí a obsahu administrátorom, - akú biznis logiku použijete je na Vás, respektíve na tom kto platí :).
15	**Databázový návrh** V databázovom návrhu treba zabezpečiť, aby databáza obsahovala: a) **minimálne 5 hlavných tabuliek**, ktoré budú reprezentovať rôzne entity systému, b) **minimálne 15 atribútov naprieč všetkými tabuľkami** (primárne a cudzie kľúče ani časové pečiatky ako created_at, updated_at sa nerátajú, ale **musia byť v DB**). Každá tabuľka musí mať logickú väzbu na ostatné. Konkrétny návrh vzťahov nechávam na vás, no musí dávať zmysel vzhľadom na funkčnosť systému a spôsob, akým spolu entity v aplikácii spolupracujú (*mali ste predmet Databázové systémy, viete tvoriť kvalitné DB návrhy*). Súčasťou odovzdania DB musia byť tri samostatné SQL súbory: - **create_script.sql** – vytvorenie celej databázy bez dát, - **demo_data.sql** – len dáta, - **db_full.sql** – kombinovaný súbor obsahujúci vytvorenie DB aj vloženie dát. Okrem toho priložte **E-R diagram vo formáte PNG exportovaný z prostredia MySQL Workbench**. Databáza musí obsahovať **dostatočný počet demo záznamov**, aby bolo možné aplikáciu po spustení okamžite otestovať/prezentovať. Potrebné minimum: aspoň 10 rôznych pízz, každá má 3 veľkosti, 15 ingrediencií, 5 tagov, 5 objednávok v rôznych stavoch (napr. pending, preparing, ready, delivering, delivered) - len teda v slovenčine. Hodnotí sa správnosť návrhu (entity, atribúty, dátové typy, väzby). **V prípade, že databáza nebude odovzdaná v spustiteľnej a funkčnej podobe alebo nebude priložený E-R diagram vo formáte PNG, celý projekt sa považuje za neodovzdaný a bude hodnotený 0 bodmi.**

15	**Java entity a ORM** Aplikácia musí používať **Spring Data JPA** a **Hibernate** pre mapovanie databázových tabuliek na Java entity. Každá "hlavná" tabuľka z databázy musí mať zodpovedajúcu entitu, pričom je potrebné správne nastaviť vzťahy medzi nimi. Mapovanie musí byť funkčné a zodpovedať logike databázového návrhu. Dbajte na bezpečnú manipuláciu s dátami, najmä pri entitách, ktoré sú súčasťou historických alebo naviazaných záznamov (napr. objednávok). Po odoslaní alebo vytvorení záznamu, ktorý na ne odkazuje, **nesmie dôjsť k zmene historických údajov**. Trochu Vás nasmerujem ako je tento problém možné vyriešiť: a) uložením vybraných hodnôt (napr. názvu, popisu, ceny) priamo do tabuľky objednávky alebo jej položiek ako textových polí bez väzby na pôvodnú entitu, b) alebo znemožnením úprav či mazania záznamov, ktoré už boli použité v objednávke alebo inom viazanom objekte (toto veľmi neodporúčam robiť pre všetky entity), c) kombinácia oboch prístupov.
15	**Trojvrstvová architektúra** Použite klasickú Controller – Service – Repository architektúru. Repository vrstva musí byť implementovaná pomocou Spring Data JPA Repository. Vytvorte minimálne **5 vlastných metód** v repozitároch. Jedna povinná metóda pre searchbar.
20	**Bezpečnosť** Implementujte autentifikáciu a autorizáciu pomocou Spring Security. Musí byť zabezpečené prihlásenie, registrácia (silné heslo), odhlásenie a zmena hesla. Každý používateľ má vlastný profil, ktorý môže upravovať. Aplikácia musí obsahovať Not Found (404), Access Denied (403), Server Error (500) stránky. **Roly a prístupové práva**: - **Anonymný používateľ** – môže si prehliadať ponuku produktov (napr. pizze, kategórie, detail produktu). - **Zákazník** – (to, čo anonym) + môže pridávať položky do košíka, vytvárať objednávky, spravovať svoj profil a zrušiť vlastnú objednávku, pokiaľ je v stave pending. - **Kuchár / manažér** – (to, čo zákazník) + vidí otvorené objednávky, môže ich „zobrať“ pre seba (ak je viac kuchárov) a mení stav objednávky pending → preparing → ready. - **Kuriér** – (to, čo zákazník) + vidí objednávky v stave ready, preberá ich (ak je viac kuriérov) a mení stav → delivering → delivered. - **Admin / majiteľ** – (to, čo zákazník a kuchár) + spravuje používateľov, priraďuje roly, schvaľuje produkty, vykonáva CRUD operácie nad všetkými entitami a má plný prehľad o systéme. Systém byť mal obsahovať aj internú notifikáciu pre kuchára/manažéra, že prišla nová objednávka (napr. vizuálne zvýraznenie, ikona, aktualizácia zoznamu bez obnovy stránky). Samozrejme úprava je možná, ak máte skúsenosť s gastrom z minulosti, môžete implemenovať takúto biznis logiku do tohto procesu.
15	**Validácia** Všetky vstupné formuláre musia mať validáciu vstupných dát (takmer všetky hlavné entitné triedy). Validačné pravidlá musia zodpovedať databáze - dĺžky, povinné polia, min/max hodnoty, kontrola formátu (v riadiacej aj servisnej vrstve). Chybové správy sa musia korektne zobrazovať v Thymeleaf šablónach. Použite hybridnú validáciu z prezentácie.

	Zobrazenie údajov na stránke
20	Implementujte Thymeleaf šablóny, ktoré budú zobrazovať údaje z databázy a umožňovať ich správu v závislosti od používateľskej roly. Šablóny musia byť prepojené s Controller a Service vrstvou a všetky zobrazené údaje musia byť v súlade s prístupovými právami jednotlivých používateľov. **Obrázky** produktov a profilové obrázky používateľov **sú povinné**. Môžu byť uložené ako externý odkaz (URL). **Ponuka produktov** musí byť **filtrovaná** a umožňovať **vyhľadávanie** podľa názvu, kategórie a/alebo značky (tagu). Výpis produktov musí byť prehľadne zobrazený s možnosťou otvoriť detail položky. Pri zobrazení detailu položky (napr. produkt, objednávka) je vhodné vytvárať odkazy vo f**ormáte s čitateľným slugom** namiesto ID čísla (napr. /pizza/margarita namiesto /pizza/5). Slug by mal byť jednoznačný a generovaný napr. z názvu objektu. **Používateľské rozhranie musí byť prehľadné a logicky rozdelené.** V šablónach sa musia zobrazovať iba tie akcie, na ktoré má daný používateľ oprávnenie (tlačidlá CRUD operácií, prehľady objednávok, správa používateľov atď.).
	BONUSY
10	Zabezpečený upload obrázkov cez systém – nahratie obrázku na server, uloženie jeho cesty do databázy a zobrazenie v používateľskom rozhraní.
10	Implementujte stránkovanie pri zobrazovaní väčších zoznamov údajov (zoznam objednávok). Stránkovanie musí správne fungovať v kombinácii s vyhľadávaním a filtrovaním dát.
10	Implementácia odosielania e-mailovej notifikácie po vytvorení objednávky, napr. potvrdenie zákazníkovi.
	Záverečné ustanovenie
	Berte to tak, že sa nachádzate v období AI, kedy nie je problém si pomôcť desiatkami nástrojov, ktoré Vám dokážu zjednoduchšiť vývoj webových aplikácií, čo výrazne šetrí čas. Téma projektu je komplexná, je náročná vzhľadom na počet kreditov predmetu, skúste sa preto na to pozrieť ako na možnosť vytvoriť projekt, korý Vám zapadne do Vášho portfólia a nie len na niečo, čo musíte urobiť kvôli kreditom.
	Termín začiatku práce na projekte: 10.11.2025 Termín odovzdania projektu: **09.01.2026 23:59** Termín hodnotenia 3 projektov Vašich spolužiakov: **16.01.2026 23:59**
	Ako projekt odovzdať?
	Odporúčaný je PUBLIC github repozitár. Ale samozrejme môžete prácu odovzdať aj ako ZIP súbor (pradepodobne kvôli firewallu EDU budete musieť odstrániť z koreňového adresára aplikácie priečinok *target*). V každom prípade LINK alebo ZIP súbor odovzdáte do **aktivity na EDU**: *Spring Boot MVC aplikácia - Pizzeria* . Výhodou github linku je možnosť nahrať link aj s veľkým predstihom (EDU máva občas navečer odstávky a ak by ste chceli na poslednú chvíľu nahrať ZIP súbor, nemusí sa Vám to podariť).