

Slovenská technická univerzita
Fakulta informatiky a informačných technológií

Tímový projekt
ASICDE

Dokumentácia inžinierskeho diela

Predmet: Tímový projekt II.

Členovia tímu:
Bc. Dominik Dancs
Bc. Tadeáš Drahovský
Bc. Adam Chmara
Bc. Gergely Lengyel
Bc. Kamil Lihan
Bc. Lukáš Mišaga
Bc. Karolína Trnovcová

Vedúci tímu: Ing. Lukáš Kohútka, PhD.

Akademický rok: 2020/21

Obsah

Obsah	2
Úvod	4
Ciele pre zimný semestier	6
Ciele pre letný semestier	6
Špecifikácia požiadaviek na celkový systém	7
Funkčné požiadavky	7
Nefunkčné požiadavky	7
Architektúra	9
Vývojové prostredie pre všetky súčasti projektu	9
Backend	12
Použité technológie	12
Moduly backendu	13
core	13
parser	15
commons-error-handling	15
commons-security	15
commons-web	15
Komunikačný modul	15
Kolaboratívny modul	16
Dátový model	16
repo	17
user	18
user_role	19
organization	20
organization_invite	21
team	22
team_membership	23
activation_token	24
token_blacklist	24
classroom	25
classroom_membership	26
assignment	27
chat_room	28
chat_member	29
chat_message	30
Frontend	31
Použité technológie	32
Služby	32
Authentication service	32

User service	32
Repository service	32
Organization service	33
Collab service	33
Chat service	33
Education service	33
Middleware	33
JWT interceptor	33
Error interceptor	33
Homepage	33
Správa používateľského účtu	34
Práca s repozitárimi	34
Dashboard organizácie	35
Edukačný modul	36
Práca s triedami	36
Komunikačný modul	37
Testovanie	38
Prílohy	1
Príloha A: API dokumentácia	1
Príloha B: Frontend Mockup	1
Príloha C: Frontend screenshoty	1
Príloha D: Špecifikácie modulov projektu	1
Príloha E: Testovanie	1

Úvod

Vývojári číslicových systémov zväčša tieto systémy navrhujú v rámci jednoduchých textových editorov ako je napríklad Notepad++. Tieto editory však častokrát nie sú dostatočne robustné a vybavené na to, aby pokryli všetku požadovanú funkcionality, ktorá je vhodná alebo žiadúca pre týchto vývojárov. Editor, ktorý by predstavoval komplexnejšie a sofistikovanejšie vývojové prostredie by mohol byť riešením pre tento problém.

ASICDE je online editor, ktorý podporuje syntaktické zvýrazňovanie textu, automatické dopĺňanie textu, kontrolu syntaktických chýb a vývoj číslicového dizajnu integrovaných čipov, konkrétnie pre programovací jazyk SystemVerilog. Toto vývojové prostredie by okrem editovania a vizualizácie kódu umožňovalo aj kolaboráciu a komunikáciu v rámci skupiny ľudí alebo organizácie a jej tímov. Vďaka týmto funkcionalitym by vývoj rozsiahlejších číslicových systémov mal byť jednoduchší, rýchlejší a efektívnejší. Systém bude taktiež poskytovať možnosti predaja a kúpy jednotlivých modulov, ktoré používateľia alebo organizácie sprístupnia k predaju v rámci obchodu s licenciami na IP jadrá.

Tento projekt sme prevzali po predošlých študentoch, ktorí sa tomuto projektu venovali v rámci diplomových prác. Naimplementovali v ňom základné funkcionality editora, ako aj základné funkcie správy používateľov na backende. Kvôli tomu, že dané funkcionality neboli kompletne implementované a odladené, niekedy nesplňali zadefinované konvencie a samozrejme, že vždy je možné systém zlepšovať, sme sa rozhodli v tomto projekte pokračovať. Našimi úlohami je ďalej vyvíjať tento systém, opraviť v ňom existujúce chyby a dokončiť neúplné funkcionality, vylepšovať ho, zvýšiť jeho bezpečnosť a pridávať do neho aj nové funkcionality.

Našou hlavnou motiváciou pre riešenie tohto projektu je práve jeho komplexnosť, zaujímavosť a vysoký potenciál byť užitočným pre mnohé firmy ako aj jednotlivcov. Ďalším dôvodom bola nedostupnosť iných dostatočne robustných nástrojov, ktoré na rozdiel od tohto systému boli často krát obmedzené, neintuitívne, ba dokonca nepoužiteľné v reálnom nasadení.

Tento dokument obsahuje informácie o systéme vytvorenom v rámci témy “Webové IDE pre ASIC [ASICDE]”. Zdokumentované sú v ňom aspekty ohľadom celkovej architektúry, ako aj samostatne časti backendu a frontendu, jednotlivé moduly a komponenty z ktorých sa skladá.

Ciele pre zimný semester

Pre zimný semester sme si stanovili nasledovné ciele:

- Rozbehanie a spustenie už existujúceho projektu
- Implementovať automatické buildenie a testovanie
- Kompletný refactoring kódu
- Upravenie konfigurácií
- Opravenie základných častí backendu - databáza, API
- Navrhnutie mockupov obrazoviek pre frontend
- Upravenie a implementovanie prvých obrazoviek (home page, login, registration)
- Spísanie dokumentácií k projektu

Celkový pohľad po zimnom semestri

Za trvania zimného semestra sa nám podarilo vytvoriť upravenú formu existujúceho systému tak, aby spĺňal všetky stanovené ciele, ktoré boli naň určené za dané obdobie.

Ciele pre letný semester

Pre letný semester sme si stanovili nasledovné ciele:

- Dokončiť refactoring kódu backendu aj frontendu
- Prispôsobiť a zlepšiť návrh aj dokumentáciu API
- Návrh a implementácia komunikačného modulu
- Dokončenie práce na implementácii organizácií a tímov
- Prerobenie všetkých služieb na API v2 (implementácia na strane backendu a prispôsobenie frontendu)
- Opravy zle implementovaných (aj chybných) častí frontendu
- Návrh a implementácia edukačného modulu
- Integrovanie nových služieb (modulov)
- Upravenie konfigurácií
- Spísanie dokumentácií k projektu
- Testovanie produktu

Celkový pohľad po letnom semestri

Za trvania letného semestra sa nám podarilo vylepšiť a rozšíriť funkcionality existujúceho systému, aby spĺňal všetky stanovené ciele, ktoré boli určené pre dané obdobie. Projekt sme taktiež pripravili pre ďalších študentov, boli odstránené nepotrebné komponenty, duplicitné súbory, funkcie a komentáre.

Špecifikácia požiadaviek na celkový systém

Funkčné požiadavky

- Používateľ musí byť prihlásený aby mohol používať aplikáciu.
- Po registrácii, pred prvým prihlásením, používateľ musí potvrdiť svoju emailovú adresu pomocou linku ktorý dostane mailom.
- Po prvom prihlásení sa zobrazí tutoriál ktorý ukáže funkcionality aplikácie a možnosti používateľa.
- Používateľ môže byť členom iba jednej organizácie, alebo môže vlastniť organizáciu.
- Vlastník organizácie môže pozvať členov do svojej organizácie podľa e-mailovej adresy alebo používateľského mena.
- Po odoslaní pozvánky registrovaní používateľ je notifikovaný aj v rámci aplikácie, aj mailom. Používateľ, ktorý ešte nemá účet je notifikovaný mailom aby sa zaregistroval.
- V organizácii sú nasledujúce roly:
 - a. Owner - vlastník organizácie (vždy iba jeden) má plné právomoci
 - b. Member - člen organizácie má oprávnenie vidieť jej obsah a môže mať prístup aj do niektorých tímov alebo repozitárov
- V rámci organizácií sa dajú vytvoriť tímy, do ktorých sa dá pridať používateľ iba taký, ktorý je členom organizácie.
- Pre tím je možné vytvoriť repozitár, na ktorom môžu pracovať iba členovia tímu.
- V rámci tímov tiež existujú role:
 - a. Team Admin - plné právomoci správy tímu (premenovanie, pridávanie členov, vytváranie a správa repozitárov)
 - b. Team Member - oprávnenie vidieť obsah tímu a modifikovať kód v repozitároch
 - c. Viewer - oprávnenie vidieť obsah tímu bez možnosti kontribúcie
- Každý používateľ má možnosť upravovať svoje osobné údaje a nastavenia účtu, a má možnosť svoj účet zmazať
- Každý používateľ má plné právomoci na svoje vlastné repozitáre, ktoré nespadajú pod žiadnu organizáciu
- Používateľ, teda vlastník repozitáru, má možnosť previesť vlastníctvo repozitára na iného používateľa

Nefunkčné požiadavky

- Odozva aplikácie musí byť menej ako 5 sekúnd
- Riešenie musí byť nezávislé od operačného systému
- Systém musí byť možné použiť viacerými používateľmi naraz
- Systém musí byť jednoducho škálovateľný
- Systém musí byť responzívny
- Systém musí byť modulárny
- Systém musí splňať prísné podmienky dostupnosti
- Systém musí byť jednoducho konfigurovateľný
- API rozhranie spĺňa konvencie a štandardy REST
- Systém musí byť zabezpečený voči neautorizovaným operáciám

- API rozhranie neumožňuje získanie citlivých údajov určených pre interné spracovanie (heslá a iné metadáta)

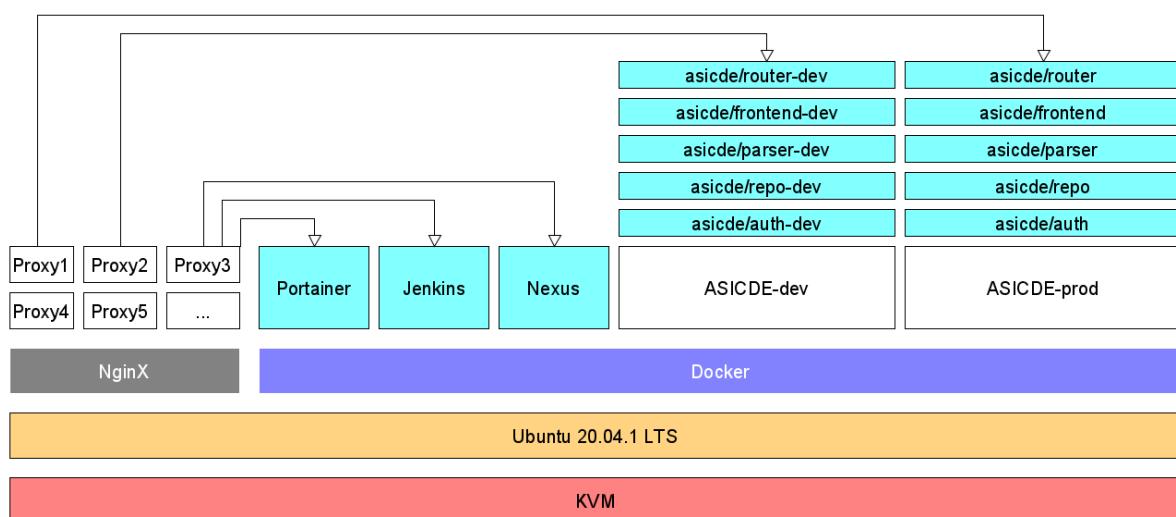
Architektúra

Našu aplikáciu vieme rozdeliť na dve hlavné časti:

- Backend
 - Základná časť, ktorá je implementovaná vo frameworku Spring (programovací jazyk Java)
 - Komunikačný a kolaboratívny modul sú implementované v jazyku TypeScript.
- Frontend - je implementovaná vo frameworku Angular (programovací jazyk JavaScript / TypeScript)

Vývojové prostredie pre všetky súčasti projektu

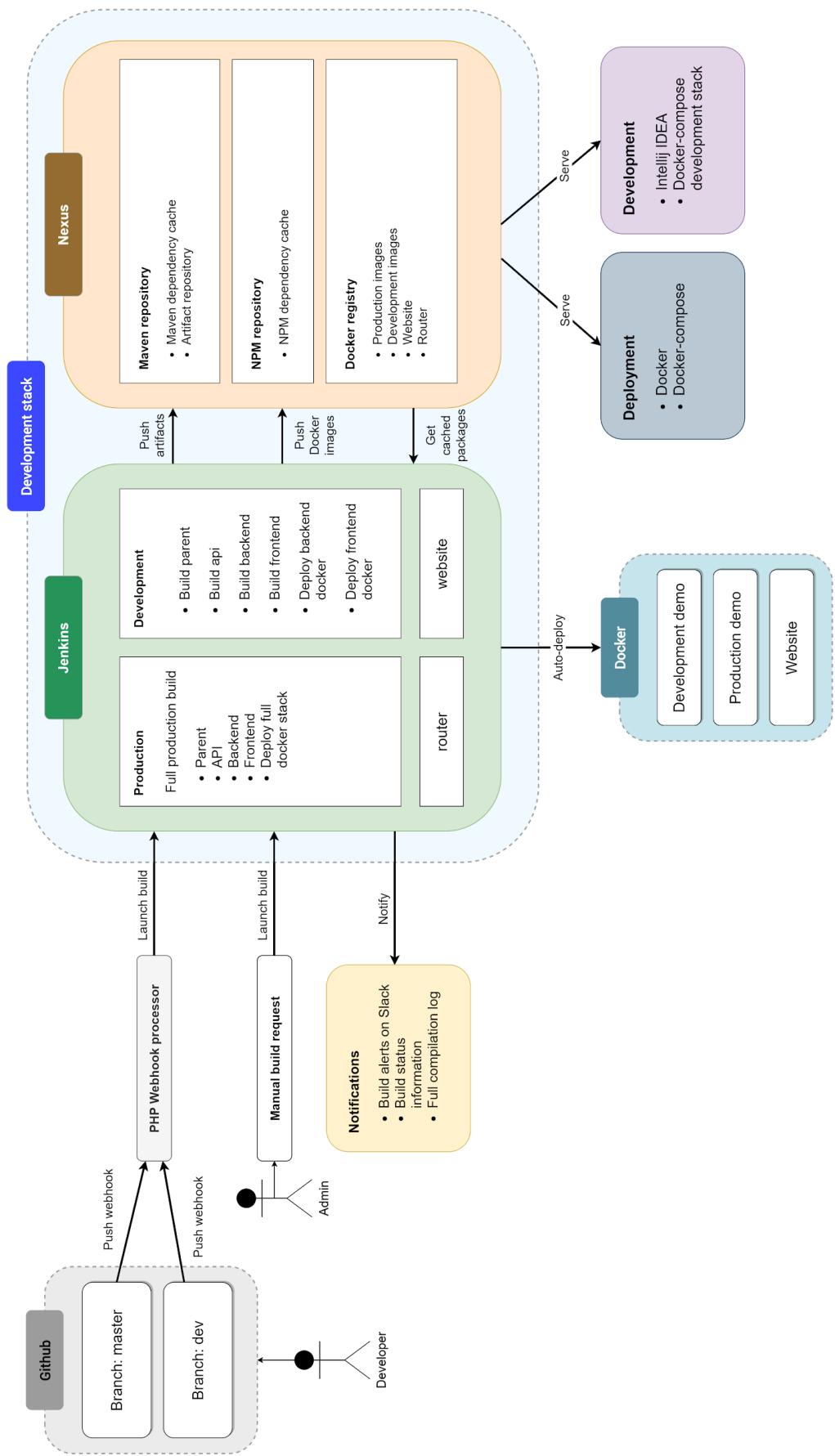
Kvôli komplexnosti projektu a jeho nasadeniu sme si vytvorili automatizované prostredie, ktoré nám uľahčí vývoj, umožní nám odsledovať všetky chyby pri kompliacii kódu ako aj nasadiť aplikáciu bez potreby manuálneho zásahu. Celý kód tohto projektu sa taktiež nachádza na kolaboračnej platforme GitHub.



Využili sme pre to virtuálny stroj, ktorý nám poskytla fakulta v rámci riešenia tohto tímového projektu. Virtuálny stroj je nainštalovaný s operačným systémom Ubuntu 20.04.1 LTS. Na tento stroj sme následne nainštalovali platformu Docker a webový server a proxy Nginx. Na platforme Docker sme nasadili jednotlivé komponenty potrebné pre automatizovaný vývoj a nasadenie ako aj samotné inštancie našej aplikácie. Webový server a proxy nám umožňuje kontrolovať prístup k aplikáciám ako aj smerovať rôzne webové adresy na služby, ktoré využívame. Služba Portainer nám umožňuje sledovať stav a spravovať jednotlivé služby nasadené v rámci platformy Docker.

Medzi služby potrebné pre vývoj patria Jenkins a Nexus. Jenkins je služba ktorá umožňuje spúšťanie automatizovaných úkonov. Využívame ju na spúšťanie kompliacie kódu hned po jeho aktualizácii na platforme GitHub, kde je náš kód uchovávaný. Jenkins sleduje celý priebeh kompliacie a testovania jednotlivých modulov a odosiela notifikácie o úspešnosti kompliacií na náš tímový Slack (náš komunikačný prostriedok). V prípade, že sa v kóde

vyskytnú chyby a komplilácia skončí s chybou, vieme na takúto situáciu promptne reagovať a dané chyby opraviť. Nexus je repozitár artefaktov, knižníc, balíkov a Dockerových obrazov. Slúži ako server pre uchovanie rôznych knižníc a balíkov, ktoré využívame v našom projekte v pamäti cache, aby ich nebolo potrebné pri každej komplilácii stiahovať z internetu. Okrem toho slúži na uchovanie medzivýsledkov komplilácie a hotových modulov v rámci nášho projektu, čo nám umožňuje zdieľať tieto súbory medzi všetkými programátormi. Jenkins taktiež vytvára na konci úspešnej komplilácie kód obrazy (balíky) pre platformu Docker, ktoré je následne možné nasadiť na akýkoľvek stroj, kde je Docker nainštalovaný, a tieto obrazy sú uložené taktiež na serveri Nexus.



Backend

Backend je súčasť nášho systému, ktorá je zodpovedná za obsluhovanie požiadaviek z používateľského rozhrania. Hlavnými úlohami backendu sú autentifikácia a autorizácia používateľov, správa súborov, správa používateľských účtov, organizácií, tímov a repozitárov.

Táto časť je implementovaná formou API (aplikačného programovacieho rozhrania), ktorého dokumentácia je súčasťou tohto dokumentu a zobrazuje všetky volania, ich vstupy, výstupy a chybové kódy, ktoré môžu požiadavky a odpovede nadobudnúť - Príloha A.

Kedže systém je navrhnutý tak, aby bol modulárny a škálovateľný, jednotlivé časti backendu sú oddelené tak, aby bola táto požiadavka splnená a jednotlivé moduly nemali medzi sebou návaznosti. Jednotlivé moduly systému a ich funkcionality sú opísané v sekcií Moduly backendu.

Použité technológie

V rámci backendu bol použitý programovací jazyk Java, nad ktorým je postavený framework Spring, ktorý slúži na implementovanie robustného API rozhrania. V projekte využívame aj správcu balíkov a verzií Maven. Vďaka tomuto nástroju dokážeme jednoducho pracovať s nadväzujúcimi knižnicami, ktoré sú potrebné v projekte, ako aj jednoducho zadefinovať spôsoby komplilácie našej aplikácie, tvorby artefaktov, ukladanie pomocných súborov do pamäte cache alebo formy testovania aplikácie.

Používané verzie nástrojov:

- Java - OpenJDK 13
- Maven - Apache Maven 3.6.3
- Spring - 2.2.5
- PostgreSQL 12

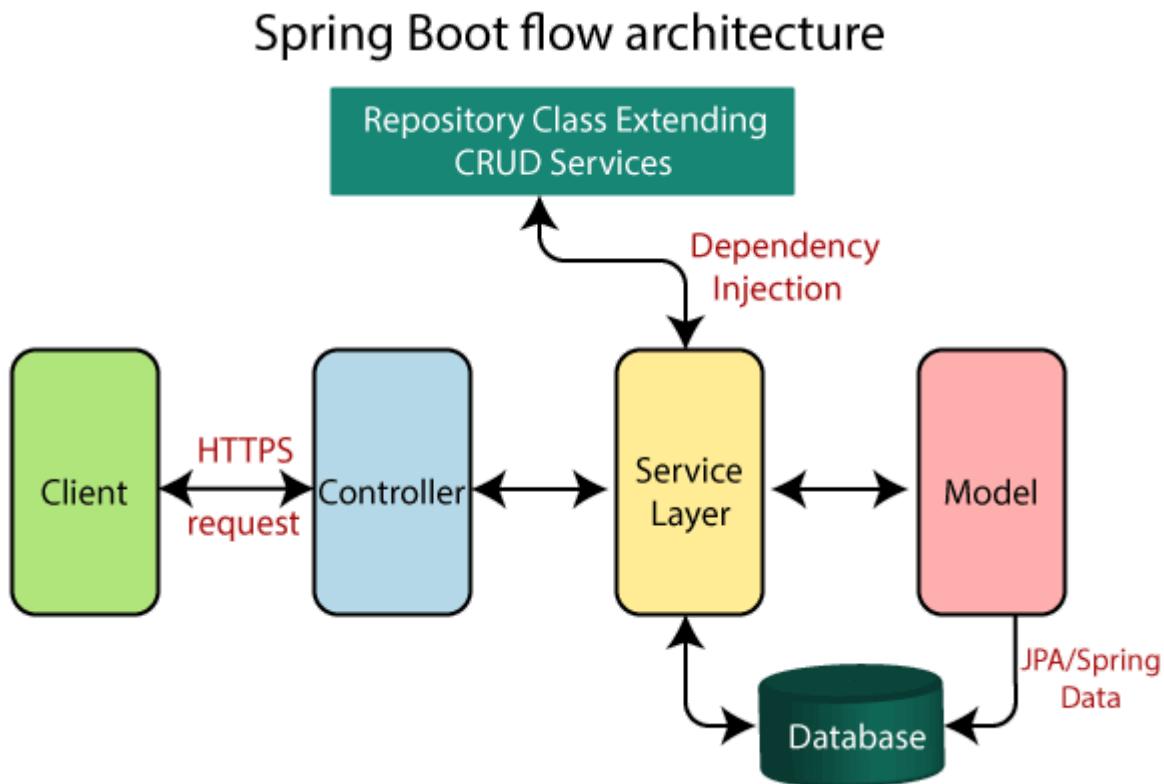
Používané verzie knižníc:

- lombok - 1.18.16
- postgresql - 42.2.5
- liquibase-core - 3.8.5
- modelmapper - 2.3.9
- jackson-jsog - 1.1.1
- java-jwt - 3.11.0

Na vývoj sme používali vývojové štúdio (IDE) IntelliJ Idea (Community edition), keďže tento nástroj poskytuje priamu podporu jazyka Java a poskytuje možnosť nainštalovať si rôzne doplnky pre prácu s frameworkom Spring ako aj knižnicami Lombok či Maven. Okrem užitočných doplnkov a šikovného prostredia poskytuje aj nástroje na odlaďovanie chýb.

Moduly backendu

V tejto časti opíšeme niektoré dôležitejšie moduly a ich úlohy. Všeobecná štruktúra modulu zodpovedá konvenciam vo frameworku Spring, kde sa každý modul skladá z troch časťí (MVC): dátový model, kontrolér a zobrazenie. V realite ide o dátovú vrstvu, servisnú vrstvu a prezentačnú vrstvu.



core

Core modul je hlavný modul našej aplikácie. Tento modul obsahuje implementáciu nasledujúcich funkcia:

- Správa používateľov, autentizácia, autorizácia, vytváranie používateľských účtov
- Správa organizácií a tímov v rámci organizácie
- Vytváranie a spravovanie repozitárov
- Komunikačný (chat) modul - API
- Edukačný modul - správa tried, zadanií
- Správa projektových súborov

Tento modul je v prípade potreby možné použiť aj v inej implementácii webového IDE, nakoľko z pohľadu dependencií je minimálne zviazaný s ostatnými súčasťami projektu. Modul obsahuje balíky rozdelené na základe použitia (konfiguračné súbory, perzistenčná vrstva, servisná vrstva, webová vrstva, zabezpečenie...).

Poskytované API týmto modulom je zabezpečené pomocou JWT (*Json Web Token*), a takmer ku všetkým dopytom na toto API je potrebný práve spomenutý JWT - tým je zaručená

identifikácia a autorizácia používateľa, ktorý daný dopyt vykonal. Dokumentácia API je priložená v tejto práci.

Všeobecný tok spracovania dopytu je nasledujúci:

1. Prijatie dopytu na aplikačnej vrstve (Controller), validácia a extrakcia tela a parametrov dopytu
 - a. Ošetrenie autorizácie - JWT a zároveň z pohľadu domény (PreAuthorize anotácia)
2. Volanie servisnej vrstvy (Service, resp. Service implementation)
 - a. Ošetrenie korektnosti parametrov (nulové hodnoty a pod.)
3. Získanie dát z perzistenčnej vrstvy (spravované pomocou JPA/Hibernate) - v prípade nutnosti, ako napr. ukladanie objektu, získanie objektu resp. dát a podobne
4. Mapovanie objektu do API odpovede (mapovanie z pohľadu entity na Data Transfer Object (DTO))
5. Odoslanie odpovede pomocou HttpServlet-u

V prípade neočakávaného stavu alebo neúspešnej autentizácie/autorizácie po spracovaní dopytu ošetrujeme tieto situácie pomocou špecifických HTTP kódov a vlastnej chybovej hlášky, ktorá je obsiahnutá v odpovedi na daný dopyt.

Dátová vrstva pozostáva z PostgreSQL databázy. Databáza je rozdelená na viacero schém:

- public - základná schéma PostgreSQL databázy. V tejto schéme sa nachádzajú tabuľky potrebné pre verziovací systém Liquibase
- chat - schéma pozostávajúca z databázových tabuľiek pre komunikačný modul
- core - schéma obsahujúca tabuľky zodpovedajúce entitám pre modul *core*

Ako už bolo spomenuté vyššie, na verziovanie databázy používame technológiu Liquibase, ktorá zabezpečuje vytváranie databázových objektov a ich úpravu pomocou štruktúrovanej schémy (v tomto projekte vo formáte XML), a tým môžeme abstrahovať od používania samotných SQL databázových dopytov - avšak v prípade nutnosti môžeme tiež vytvoriť vlastný SQL skript, ktorý bude vykonávaný ako každý iný verziovací súbor.

Konfigurácia core modulu je možná pomocou premenných prostredia, alebo pomocou premenných definovaných v súbore *resources/application.properties*. Všetky špecifické nastavenia pre tento projekt sú nasledovné:

```
app.jwt.secret=JWTSuperSecretKey
app.jwt.expiration.time=3600
auth.url=http://core:8080/api/auth
app.account.verify=false
app.storage.location=/data
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=500MB
spring.servlet.multipart.max-request-size=500MB
spring.datasource.url=jdbc:postgresql://db:5432/asicde?currentSchema
=public
spring.datasource.driverclassname=org.postgresql.Driver
spring.datasource.username=asicde
spring.datasource.password=password
```

```
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.generate_statistics=false
```

parser

Parser modul rieši prácu s kódom ktorý používatelia upravujú v editore aplikácie. Jeho hlavnou úlohou je analýza a validácia tohto kódu.

commons-error-handling

Modul commons-error-handling implementuje vlastnú logiku mapovania chybových hlášok na odpovede obsiahnuté v REST odpovediach. Je knižnica obsahujúca definíciu projektových výnimiek, ktoré sa používajú v každom module. Každá nízko úrovňová výnimka vyhodená knižnicou tretej strany, je zaobalená do jednej z projektových výnimiek definovanej práve v tejto knižnici. Obsahuje tiež konfiguráciu REST API, ktorá každú projektovú výnimku preloží do HTTP odpovede s prislúchajúcim HTTP statusom.

commons-security

Je knižnica zodpovedná za zabezpečenie aplikácie. Obsahuje konfiguračné súbory, ktoré nastavujú a dopĺňajú základnú Spring Security vrstvu. Definuje úroveň zabezpečenia pre jednotlivé cesty REST služby, obsahuje v sebe Autentifikačného klienta, pomocou ktorého priamo komunikuje s autentifikačným modulom. Ak je potrebné, aby nový modul podporoval zabezpečenie, stačí importovať knižnicu Commons Security a načítať konfiguračné súbory do kontextu aplikácie. Tento modul momentálne nie je plne využívaný v našej práci, nakoľko v dôsledku rozsiahlych zmien sa presunula logika autentizácie používateľov do modulu *core*.

commons-web

Obsahuje v sebe konfiguračné súbory, ktoré nastavujú webovú vrstvu aplikácie. Importuje Spring Web knižnicu aj s vybranými konfiguračnými súbormi. Konfiguruje Spring Actuator, ktorý umožňuje sledovať základné údaje o aplikácii ako status a rôzne metriky počas behu. Tento modul by mala využiť každá súčasť systému, ktorá pracuje tiež na úrovni webových (REST) služieb. Okrem už spomenutých funkcionálít tento modul obsahuje tiež konfiguráciu, ktorá riadi spracovanie výnimiek (Exceptions), ktoré sa ďalej spracúvajú v komponente *commons-security*.

Komunikačný modul

Komunikačný modul je oddelený od hlavného modulu projektu. Je napísaný v jazyku TypeScript bez podporného frameworku. Tento modul slúži na sprostredkovanie rozhrania medzi samotnou implementáciou na frontende a hlavným modulom, ktorý slúži na správu a ukladanie komunikácie. Komunikačný backend disponuje websocket serverom, ktorý umožňuje komunikáciu v reálnom čase a následne komunikuje s API na hlavnom backendu. Na vytvorenie tohto komunikačného kanálu je využitá knižnica Socket.io. Úlohou tohto modulu je vytvárať a manažovať chatovacie miestnosti, pripojených klientov, a distribuovať odosielané správy.

Kolaboratívny modul

Tento modul je prácou ďalšieho študenta a je zapojený do systému ako izolovaný modul podobne ako komunikačný modul. Jeho úlohou je sprostredkovať funkciu zdieľaného editovania zdrojových kódov viacerými používateľmi súčasne. Pre komunikáciu so zvyškom systému je rovnako využívané API rozhranie hlavného modulu.

Dátový model



repo

▼ Fields (12)

Name	Type	Settings	References
id	bigint	not_null  PK increment	↳ team.repo_id ← chat_room.repo_id
uuid	uuid	not_null unique	
name	varchar(50)	not_null	
uri	text	not_null	
author_uuid	uuid	not_null	↳ user.uuid
storage_name	varchar(50)	not_null	
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		
favorite	boolean(default: FALSE)		
description	varchar(255)		

USER

▼ Fields (14)

Name	Type	Settings	References
id	bigint	<code>not_null</code>  <code>increment</code>	← user_role.user_id ← team_membership.user_id ← activation_token.user_id +5 more
uuid	uuid	<code>not_null</code> <code>unique</code>	← repo.author_uuid
email	varchar(50)	<code>not_null</code> <code>unique</code>	
username	varchar(50)	<code>not_null</code> <code>unique</code>	
first_name	varchar(50)	<code>not_null</code>	
last_name	varchar(50)	<code>not_null</code>	
password	varchar(255)	<code>not_null</code>	
active	boolean	<code>not_null</code>	
description	varchar(255)		
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		
organization_id	bigint		→ organization.id

user_role

▼ Fields (3)

Name	Type	Settings	References
id	bigint	not_null  PK increment	
user_id	bigint	not_null	↳ user.id
role	varchar(50)	not_null	

▼ Indexes (1)

Columns	Name	Type	Settings
user_id			

organization

▼ Fields (8)

Name	Type	Settings	References
<code>id</code>	bigint	<code>not_null</code> <code>PK</code> <code>increment</code>	↳ organization_invite.organization_id ↳ team.organization_id ↳ chat_room.organization_id +1 more
<code>uuid</code>	uuid	<code>not_null</code> <code>unique</code>	
<code>name</code>	varchar(100)	<code>not_null</code>	
<code>owner_id</code>	bigint	<code>not_null</code>	→ user.id
<code>created</code>	timestamp		
<code>created_by</code>	varchar(255)		
<code>last_modified</code>	timestamp		
<code>last_modified_by</code>	varchar(255)		

▼ Indexes (1)

Columns	Name	Type	Settings
<code>owner_id</code>			

organization_invite

▼ Fields (9)

Name	Type	Settings	References
id	bigint	not_null  PK increment	
uuid	uuid	not_null unique	
email	varchar(255)	not_null	
status	varchar(255)		
organization_id	bigint	not_null	↳ organization.id
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		

▼ Indexes (1)

Columns	Name	Type	Settings
organization_id			

team

▼ Fields (9)

Name	Type	Settings	References
id	bigint	not_null PK increment	← team_membership.team_id ← chat_room.team_id
uuid	uuid	not_null unique	
name	varchar(50)	not_null	
organization_id	bigint		→ organization.id
repo_id	bigint		← repo.id
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		

▼ Indexes (1)

Columns	Name	Type	Settings
repo_id			

team_membership

▼ Fields (9)

Name	Type	Settings	References
id	bigint	not_null  PK increment	
uuid	uuid	not_null unique	
user_id	bigint	not_null	↳ user.id
team_id	bigint	not_null	↳ team.id
permission	varchar(63)	not_null	
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		

▼ Indexes (2)

Columns	Name	Type	Settings
user_id			
team_id			

activation_token

▼ Fields (6)

Name	Type	Settings	References
<code>id</code>	<code>bigint</code>	<code>PK</code> increment	
<code>created</code>	<code>timestamp</code>		
<code>last_modified</code>	<code>timestamp</code>		
<code>uuid</code>	<code>varchar(255)</code>	unique	
<code>token</code>	<code>varchar(64)</code>	not_null	
<code>user_id</code>	<code>bigint</code>	not_null	↳ user.id

token_blacklist

▼ Fields (4)

Name	Type	Settings	References
<code>id</code>	<code>bigint</code>	<code>PK</code> increment	
<code>uuid</code>	<code>varchar(255)</code>	unique	
<code>token</code>	<code>varchar(4096)</code>	not_null	
<code>expiration</code>	<code>timestamp</code>		

▼ Indexes (1)

Columns	Name	Type	Settings
<code>expiration</code>			

classroom

▼ Fields (10)

Name	Type	Settings	References
id	bigint	PK increment	← classroom_membership.classroom_id ← assignment.classroom_id
name	varchar(100)	not_null	
uuid	uuid	not_null unique	
slug	varchar(255)	not_null unique	
password	varchar(255)	not_null	
lock	boolean(default: FALSE)		
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		

classroom_membership

▼ Fields (9)

Name	Type	Settings	References
id	bigint	PK increment	
uuid	uuid	not_null unique	
user_id	bigint	not_null	→ user.id
classroom_id	bigint	not_null	→ classroom.id
role	varchar(63)	not_null	
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		

▼ Indexes (2)

Columns	Name	Type	Settings
user_id			
classroom_id			

assignment

▼ Fields (15)

Name	Type	Settings	References
id	bigint	PK increment	↳ assignment.parent_id
uuid	uuid	unique	
name	varchar(50)	not_null	
author_id	bigint	not_null	↳ user.id
parent_id	bigint		↳ assignment.id
classroom_id	bigint	not_null	↳ classroom.id
storage_name	varchar(50)	not_null	
published	boolean(default: FALSE)		
due_date	timestamp		
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		
favorite	boolean(default: FALSE)		
description	varchar(255)		

▼ Indexes (3)

Columns	Name	Type	Settings
classroom_id			
author_id			
parent_id			

chat_room

▼ Fields (13)

Name	Type	Settings	References
id	bigint	not_null PK increment	← chat_member.chat_room_id ← chat_message.chat_room_id
uuid	uuid	not_null unique	
created	timestamp		
created_by	varchar(255)		
last_modified	timestamp		
last_modified_by	varchar(255)		
name	varchar(255)		
should_expire_in	int	not_null	
last_active	timestamp		
owner_id	bigint		↳ user.id
organization_id	bigint		↳ organization.id
team_id	bigint		↳ team.id
repo_id	bigint		↳ repo.id

▼ Indexes (4)

Columns	Name	Type	Settings
owner_id			
organization_id			
team_id			
repo_id			

chat_member

▼ Fields (6)

Name	Type	Settings	References
<code>id</code>	<code>bigint</code>	<code>not_null</code>  <code>PK</code> <code>increment</code>	← chat_message.chat_member_id
<code>uuid</code>	<code>uuid</code>	<code>not_null</code> <code>unique</code>	
<code>name</code>	<code>varchar(255)</code>		
<code>chat_room_id</code>	<code>bigint</code>	<code>not_null</code>	→ chat_room.id
<code>user_id</code>	<code>bigint</code>	<code>not_null</code>	→ user.id
<code>activity_status</code>	<code>varchar(63)</code>		

▼ Indexes (1)

Columns	Name	Type	Settings
<code>user_id</code>			

chat_message

▼ Fields (7)

Name	Type	Settings	References
id	bigint	not_null  PK increment	
uuid	uuid	not_null unique	
message	text		
author	varchar(255)		
timestamp	timestamp		
chat_member_id	bigint	not_null	↳ chat_member.id
chat_room_id	bigint	not_null	↳ chat_room.id

▼ Indexes (1)

Columns	Name	Type	Settings
chat_room_id			

Frontend

Frontend je používateľské rozhranie pre tento systém. Sprístupňuje hlavnú stránku projektu ako aj rôzne funkcionality editora, organizácií, či chatu. Na pozadí komunikuje s API rozhraním na backende, vďaka ktorému môže registrovať, autentifikovať a autorizovať používateľov, zobrazovať im ich repozitáre, informácie o používateľskom účte, a spravovať organizácie či tímy. Hlavné funkcie editora, ako vizualizácia kódu, oprava syntaxe či validácia písaného kódu sa z väčšej časti realizuje priamo v klientskej aplikácii. Backend slúži hlavne na uchovanie súborov repozitárov a ďalších dát.

Samotný frontend sa skladá z viacerých modulov, ako napríklad auth - login, registrácia a autentifikácia používateľa, editor - práca s editorom, home - nastavenia používateľa a repozitárov, app-routing.module - prepája stránky na frontende, a ďalšie. Tieto moduly sú zväčša rozdelené do viacerých komponentov. Každý z komponentov sa skladá z implementácie layoutu v HTML, definovania štýlu v SCSS, testov napísaných v TypeScripte a samotného controlleru taktiež napísanom v TypeScripte. Okrem modulov obsahujúcich komponenty sú na frontende implementované viaceré services, čiže služby, ktoré slúžia na komunikáciu s backendom, alebo spracovanie dát.

V aktuálnom stave sú implementované nasledovné funkcionality:

- Webové rozhranie - využitie frameworku Angular a Typescript.
- Editor - založený na existujúcom Monaco editore, s pridanou podporou programovacieho jazyka SystemVerilog.
- Editor - podporuje syntaktické zvýrazňovanie textu, kontrolu syntaktických chýb a automatické dopĺňanie textu.
- Editor - je doplnený kolaboračným modulom pre spoluprácu viacerých používateľov v rámci jedného repozitáru
- Komunikácia s API rozhraním (backend).
- Homepage - úvodná stránka do systému. V prípade, že používateľ nie je prihlásený, je mu ponúknutá možnosť prihlásenia alebo registrácie.
- Autorizácia - prihlásenie existujúceho používateľa alebo registrácia nového používateľa.
- Autentifikácia - používateľ je autentifikovaný pomocou JWT (JSON web token).
- Správa používateľa - zahŕňa spravovanie osobných údajov.
- Správa repozitárov - zahŕňa zobrazenie a spravovanie všetkých osobných repozitárov.
- Úprava nastavení (osobných a repozitárov) je povolená až po úspešnom prihlásení používateľa.
- Organizácie - používateľ vie vytvoriť organizáciu. Používateľ, ktorý organizáciu vytvoril, je jej vlastníkom. Vlastník organizácie ju následne vie aj zmazať.
- Správa členov organizácie - Vlastník organizácie vie pridať členov do organizácie, aj ich z organizácie odstrániť.
- Správa tímov - v rámci organizácie používateľ vie vytvoriť tímy. Do existujúceho tímu môžu byť priradení členovia alebo môžu byť z tímu odstránení. Členom tímu môžu byť zmenené ich práva.

Použité technológie

Klientská časť systému bola implementovaná v jazykoch JavaScript a TypeScript za pomocí framework AngularJS. V rámci tohto frameworku sme využili aj nasledovné dodatočné moduly, ktoré nám sprístupňujú dôležité funkcionality:

- angular/core - 9.1.6
- angular/common - 9.1.6
- angular/material - 9.2.3
- angular/router - 9.1.6
- ng-bootstrap/ng-bootstrap - 8.0.0
- syncfusion/ej2-angular-navigations - 18.1.45
- jszip - 3.4.0
- ngx-monaco-editor - 8.1.1
- socket.io-client - 4.0.1
- zip-js - 0.0.2

Na implementovanie samotného prostredia pre písanie kódu je namiesto implementovania vlastného editora použitý existujúci open-source editor Monaco. Tento editor disponuje funkcionalitami ako napríklad syntax highlighting, vyznačovanie chýb a upozornení a pridávanie nových jazykových modulov. Toto nám veľmi uľahčilo prácu a okrem toho je možné tento editor nezávisle od modifikácií, ktoré doňho budú nami implementované, aktualizovať.

Služby

Authentication service

Autentifikačná a autorizačná služba je kritickým komponentom celého systému. Slúži na prvotné prihlásenie a registráciu používateľa ako aj na udržanie aktuálnej relácie používateľa v rámci aplikácie. Jej hlavnou úlohou je správa relácie používateľa a udržiavanie tejto relácie počas celej interakcie s aplikáciou. Táto relácia je ukladaná v rámci local storage v prehliadači a pri interakcii so systémom sa nepretržite obnovuje, aby používateľ nestrelil prístup a neboli odhlásený počas tejto interakcie. Pri odhlásení taktiež korektne zruší používateľskú reláciu a odstráni citlivé dátá, ktoré sú uchovávané v prehliadači pre správny chod aplikácie.

User service

Táto služba slúži na správu používateľov. Sprostredkúva rozhranie medzi API na backende a samotnou aplikáciou.

Repository service

Služba správy repozitárov sprostredkúva dôležitú funkciu spojenú s manažmentom a interakciou s repozitárimi. Pri vybraní repozitáru si uchováva jeho dátá, a následne umožňuje interakciu s ním. Poskytuje funkciu pre úpravu repozitára, jeho nastavení, ako aj prácu s gitom a súbormi.

Organization service

Táto služba poskytuje všetkú funkciu spojenú so správou a interakciou s organizáciami, tímami a ich repozitárimi. Umožňuje spravovať nastavenia organizácie, pridávať a odoberať jej členov, či spravovať tímy v rámci danej organizácie.

Collab service

Kolaboračná služba je kritickým komponentom systému, ktorý je úzko naviazaný na prácu v editore a prácu so súbormi. Jeho hlavou funkciou je sprostredkovanie interakcie s repozitárom a jeho súbormi v reálnom čase, teda vytvára medzi vrstvu spracúvajúcu zmeny v zdrojových kódoch a tieto zmeny následne distribuuje všetkým pripojeným klientom, a zaručuje korektné ukladanie zmien všetkých klientov. Okrem kódu repozitára rovnako synchronizuje a sprostredkuje interakciu v reálnom čase s vizualizáciou logických členov. Ďalšou funkciou je správa aktuálne pripojených klientov a zobrazovanie ich prítomnosti, pozície v editore, a ostatných interakcií.

Chat service

Komunikačná služba sprostredkováva funkciu potrebnú pre správu chatovacích miestností, ich používateľov a samotnej komunikácie, ktorá je v reálnom čase spracovávaná.

Education service

Táto služba poskytuje všetku funkciu spojenú so správou a interakciou s používateľmi tried, žiakov a zadanií. Umožňuje spravovať nastavenia triedy, pridávať a odoberať jej členov, či spravovať zadania v rámci danej triedy.

Middleware

JWT interceptor

Tento pomocný kód slúži na autorizovanie požiadaviek z aplikácie voči API rozhraniu na backende. Každá požiadavka (okrem požiadaviek na verejné časti API) musí byť autorizovaná. Na toto sú využívané JWT tokeny, ktoré sa pomocou tohto interceptora pridajú do hlavičiek požiadaviek a autorizujú používateľské interakcie.

Error interceptor

Tento pomocný kód slúži na spracovanie chybových odpovedí z API rozhrania a transformovanie chýb do správ čitateľných a zrozumiteľných pre bežného používateľa aplikácie.

Homepage

Domovská stránka predstavuje prvý kontakt s používateľom. Na nej je používateľovi poskytnutá možnosť registrácie, pokial ešte nemá založený účet. V prípade, že používateľ už je registrovaný, môže sa z domovskej stránky priamo prihlásiť. Pri zvolení jednej z dvoch možností, používateľ bude presmerovaný na príslušný formulár, kde vyplní buď svoje prihlásovacie údaje (v prípade loginu) alebo svoje osobné údaje potrebné na registráciu. Či

už v prípade registrácie, alebo prihlásenia, používateľovi je v oboch prípadoch poskytnuté prejsť na opačnú možnosť (v regisračnom formulári prejsť na login a z loginu možnosť prejsť na registráciu). V prípade, že požívateľ je prihlásený, domovská stránka ho privíta a poskytne mu možnosť prejsť na prehľad svojich repozitárov. Domovská stránka a s ňou súvisiace registrácia a prihlásenie používateľa obsahuje nasledovné komponenty:

- **LandingComponent**: Komponent domovskej stránky, ktorý slúži na to, aby sa zistilo, či je aktuálne prihlásený používateľ alebo nie je. Podľa toho bud' privíta prihláseného používateľa alebo mu poskytne možnosť prihlásenia a registrácie. Následne presmeruje používateľa na stránku jeho voľby.
- **AuthContainerComponent**: Komponent, ktorý slúži na zobrazenie, formátovanie a následne presmerovanie na stránky registrácie alebo prihlásenia.
- **LoginComponent**: Komponent slúžiaci na prihlásenie používateľa, na prijatie a kontrolu prihlásovacích údajov. V prípade, ak údaje boli nesprávne, respektíve používateľ s danými prihlásovacími údajmi neexistuje, bude vypísaná chyba oznamujúca neúspešné prihlásenie. V opačnom prípade používateľ bude úspešne prihlásený.
- **RegisterComponent**: Komponent zodpovedný za registráciu nového používateľa. V prípade, že boli zadané všetky potrebné údaje v správnom formáte, používateľ bude úspešne zaregistrovaný a systém ho presmeruje na prihlásovaci stránku, kde sa môže prihlásiť. V opačnom prípade, ak údaje neboli všetky vyplnené alebo nespĺňajú stanovený formát, používateľ bude upozornený.

Správa používateľského účtu

UserEditionComponent slúži na to, aby prihlásený používateľ mal možnosť spravovať svoj účet (My account). V rámci tohto je mu poskytnutá možnosť zmeniť si svoje meno (krstné meno a priezvisko), mailová adresa, ako aj zmeniť si heslo. Používateľ vie svoj účet aj odstrániť zo systému. V prípade, že používateľ je vlastníkom organizácie, tak v rámci týchto nastavení mu je umožnené danú organizáciu odstrániť. Taktiež, ak je používateľ členomnejakej organizácie, v nastaveniach účtu má možnosť opustiť organizáciu.

Práca s repozitármi

Ďalším prvkom na frontende je prehľad repozitárov prihláseného používateľa. Rozloženie tejto obrazovky sme koncipovali podľa predom pripravenych a konzultovaných návrhov (mockupy v prílohe). Vďaka tejto obrazovke má používateľ prístup ku svojim repozitárom, ktoré môže spravovať alebo priamo k nim pristupovať a editovať ich. V prehľade má k dispozícii tie najpodstatnejšie informácie o repozitároch ako je ich názov, popis repozitáru a pod.

- **Dashboard repozitárov**: MyReposComponent je komponent, ktorý slúži na základný prístup k repozitárom - zobrazenie repozitárov prihláseného používateľa, možnosť pridať nový repozitár, pridať repozitár medzi oblúbené, archivovať repozitár a zmazať repozitár. Umožňuje presmerovanie k nastaveniam zvoleného repozitára a otvorenie repozitára v editore.
- **Nastavenia repozitárov**: V rámci nastavení repozitára (RepositoryEditionComponent) používateľ vie upravovať zvolený repozitár - zmeniť

mu názov a URL. Používateľ by mal vedieť vykonávať ďalšie akcie, ako sú odstránenie a archivovanie zvoleného repozitára, deaktivovať synchronizáciu s gitom, zdieľať repozitár alebo zmeniť vlastníka.

- **Editor:** EditorComponent je hlavný komponent editora. V rámci tohto komponentu sú implementované základné operácie pre prácu so súbormi repozitára, navigáciu v rámci editora ako aj samotná inicializácia Monaco editora. Navigácia medzi otvorenými oknami editora a samotná implementácia editora sú oddelené do osobitných komponentov - EditorTabsComponent a EditorTileComponent.

Dashboard organizácie

Manažment organizácií je nová funkcia, ktorá bola pridaná do systému. Používateľovi je umožnené vytvoriť organizáciu, do ktorej môžu byť pozvaný členovia a môžu v nej byť vytvorené tímy a repozitáre. Správa tímov a ich členov, repozitárov a ich nastavení je zakomponovaná do časti organizácií. Používateľ má možnosť prehliadať jednotlivé repozitáre a priraďovať ich jednotlivým tímom. V časti spravovanie tímov, vidí výpis členov tímu, môže meniť ich poverenia a priraďovať jednotlivé repozitáre.

- Repozitáre:
- Tímy organizácie: V rámci organizácie môžu byť vytvárané tímy. Každý tím sa skladá z aspoň jedného člena (používateľ, ktorý tím vytvoril). Členov tímu vieme rozdeliť nasledovne:
 - a. Vlastník organizácie (Owner)
 - Vytvoril tím, automaticky je jeho vlastníkom
 - Má právo zmazať tím
 - Má právo meniť nastavenia tímu
 - Má právo pridávať a odstraňovať členov do/z tímu
 - Má právo meniť práva členov tímu
 - Vidí repozitár tímu
 - Má právo upravovať kód v rámci repozitáru
 - b. Administrátor (Admin)
 - Je spravcom tímu
 - Má právo meniť nastavenia tímu
 - Má právo pridávať a odstraňovať členov do/z tímu
 - Má právo meniť práva členov tímu
 - Vidí repozitár tímu
 - Má právo upravovať kód v rámci repozitáru
 - c. Member (read and write právo)
 - Vidí tím, do ktorého patrí
 - Vidí repozitár tímu
 - Má právo upravovať kód v rámci repozitáru
 - d. Viewer (read právo)
 - Vidí tím, do ktorého patrí
 - Vidí repozitár tímu
- Členovia organizácie: Členov organizácie delíme na:
 - a. Vlastník organizácie (Owner)
 - Vytvoril organizáciu, automaticky je jej vlastníkom
 - Má právo zmazať organizáciu

- Má právo meniť nastavenia organizácie
 - Má právo zmeniť vlastníka organizácie
 - Má právo pridávať a odstraňovať členov do/z organizácie
 - Má právo vytvárať tímy v organizácii (stane sa vlastníkom tímu)
- b. Člen organizácie (Member)
- Vidí obsah organizácie
 - Vidí tímy, ktorých je členom

Edukačný modul

Edukačný modul a jeho manažment je nová funkcia, ktorá bola pridaná do systému. Používateľovi je umožnené vytvoriť triedu, do ktorej môžu byť pozvaný členovia (žiaci alebo owneri) a môžu v nej byť vytvorené úlohy, ktoré žiaci musia vypracovať. Vypracované úlohy žiakov sa tiež nahrávajú do systému. Správa tried a ich členov, úloh a ich nastavení je zakomponovaná do časti edukačného modulu. Používateľ má možnosť prehliadať jednotlivé triedy, owner (učiteľ) má možnosť odstrániť triedy alebo upraviť triedy. V časti spravovanie tried, vidí výpis členov tímu a výpis všetkých úloh v triede.

Práca s triedami

Ďalším prvkom na frontende je prehľad tried prihláseného používateľa. Rozloženie tejto obrazovky sme koncipovali podľa dohodnutej a spísanej špecifikácie podľa vedúceho tímu. Vďaka tejto obrazovke má používateľ prístup ku svojím triedam, ktoré môže spravovať alebo priamo k nim pristupovať a editovať ich. V prehľade má k dispozícii tie najpodstatnejšie informácie o úlohách a členoch.

- **Dashboard tried:** ClassroomsComponent je komponent, ktorý slúži na základný prístup k triedam - zobrazenie tried prihláseného používateľa, možnosť pridať novú triedu alebo pripojiť sa už k existujúcej. Ak je používateľ ownerom niektorej z tried, tak ju môže odstrániť alebo zmeniť jej nastavenia, avšak to už sa rieši v nasledujúcom komponente.
- **Nastavenia tried:** V rámci nastavení tried (ClassroomSettingsComponent) používateľ vie upravovať zvolenú triedu - zmeniť názov, slug a heslo. Používateľ by mal vedieť vykonávať ďalšie akcie, ako sú zdieľať triedu alebo uzamknúť triedu.
- **Trieda:** ClassroomComponent je hlavný komponent dashboardu tried. V rámci tohto komponentu sú implementované základné operácie pre prácu so v triede. Medzi možnosti patrí vytváranie úloh, pridávanie používateľov. Okrem toho je tiež možné priamo odstrániť používateľov alebo úlohy z triedy. Taktiež sú vytvorené úlohy najskôr v stave nepublikovanom. Používateľ môže rovno z tohto komponentu vybrať úlohy, ktoré publikuje.
- **Nastavenie úloh:** V rámci nastavení úloh (AssignmentComponent) je zobrazená úloha vytvorená učiteľom spolu s úlohami, ktoré vypracovali jednotliví žiaci. Tento komponent je potrebné doimplementovať o funkcia, ktorá je hodnotenie odovzdaných úloh, zmenu dátumu odovzdania, zmenu zadania úlohy. Zatiaľ je tento komponent iba akýmsi prototypom pripraveným pre ďalšiu prácu na ňom.

Komunikačný modul

Komunikačný modul je nová funkcia, ktorá pridáva možnosť chatovania v podobe dedikovaných chatovacích miestností pre jednotlivé organizácie. Každá organizácia má by default vytvorenú chatovaci miestnosť kde sú pridaní všetci členovia organizácie. Používateľ organizácie vidí na frontende zoznam všetkých chatovacích miestností v ktorých sa nachádza, spolu s ich názvom a dodatočnými informáciami ako napr. počet aktuálne pripojených užívateľov v miestnosti. Po kliknutí na chatovaci miestnosť sa používateľ pripojí a notifikuje všetkých pripojených užívateľov o svojom pripojení do miestnosti, taktiež vidí chatovaciu história danej miestnosti aj v čase keď pripojený neboli. Používateľ potom môže ľubovoľne chatovať v miestnosti v reálnom čase. Momentálne nie je na frontende možnosť vytvoriť nové chatovacie miestnosti okrem predvolenej pre organizáciu, avšak funkcia je na backende implementovaná s potenciálom pre jednoduché rozšírenie o nové funkcionality ako user management, chatroom management, súkromné správy a podobne.

Testovanie

Kedže projekt je rozdelený na dve hlavné časti - backend a frontend, správnu funkcionality týchto častí treba zväčša osobitne testovať. Preto pre obe časti existujú automatické testy, ktoré overia správnosť špecifických požiadaviek. Tieto testy sú v aktuálnom stave, že sa musia opraviť, kedže sme v rámci tohto semestra menili veľa vecí na projekte, aj testy treba meniť a prispôsobiť aktuálnej funkcionality.

Samozrejme hlavný spôsob testovania spočíval v manuálnom testovaní. Toto spočívalo v tom, že vždy keď sa implementovala nová časť, alebo upravila existujúca časť, osoba, ktorá danú zmenu spravila, ju otestovala lokálne u seba. Ak neboli nájdené žiadne chyby, tak konkrétnu zmenu vždy otestoval ešte minimálne jeden ďalší člen z tímu. Ak aj toto bolo v poriadku, mohol sa kód mernúť s hlavným kódom, kde prebla ešte posledná etapa testovania - testovanie cez Jenkins pipelines. Tieto kroky a viaceré etapy testovania nám pomohli minimalizovať chybovosť. Na backende tiež používame unit testy pomocou Mockito.

Okrem automatizovaného testovania a testovania členmi tímu sme do testovania zapojili aj študentov FIIT a FEI. Títo študenti nám produkt priebežne testovali, a výsledné protokoly z posledného testovania sú v tomto dokumente ako Príloha E.

Prílohy

Príloha A: API dokumentácia

Dokumentácia API rozhrania pre služby na backende sú dostupné na webovej stránke projektu <https://team01-20.studenti.fiit.stuba.sk/api-docs/> alebo v repozitári dokumentácie projektu vo formáte OpenAPI V3.

Príloha B: Frontend Mockup

Úvodná stránka

The screenshot shows the homepage of a website called "ASICDE". At the top, there is a navigation bar with links for "About", "Features", "Pricing", "Contact", and "TOS". Below the navigation bar, there is a large graphic on the left featuring a stylized illustration of a person standing next to a green rectangular box, with various electronic symbols like resistors and capacitors floating around them. To the right of the graphic, the text "Your IDE of choice for SystemVerilog" is displayed in bold, followed by "Try it now, for free!". Below this, there are two buttons: a teal button labeled "Join now" and a white button labeled "Log in", separated by the word "or".

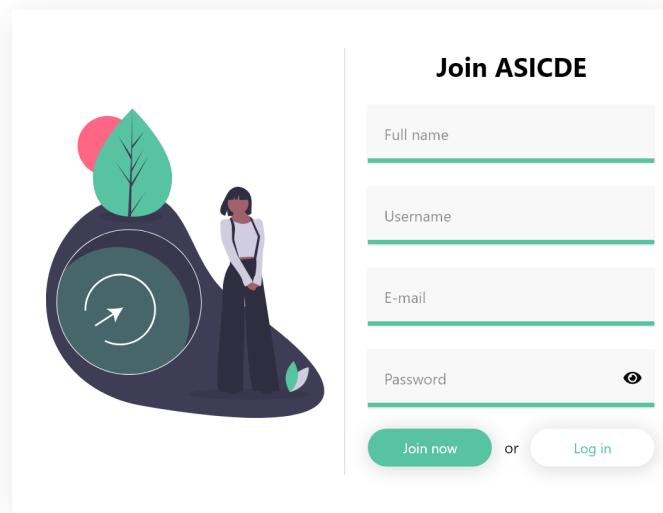
Prihlásenie

The screenshot shows the sign-in page of the ASICDE website. At the top, there is a navigation bar with links for "About", "Features", "Pricing", "Contact", and "TOS". Below the navigation bar, there is a large graphic on the left featuring a stylized illustration of a person standing in front of a house with a green door. To the right of the graphic, the text "Sign in" is displayed in bold. Below this, there are two input fields: one for "Username or e-mail" and one for "Password", which includes a visibility toggle icon. Below the password field, there are links for "Remember me" and "Forgot password?". At the bottom, there are two buttons: a teal button labeled "Login" and a white button labeled "Join now", separated by the word "or".

Registrácia

ASICDE

About | Features | Pricing | Contact | TOS



Domovská stránka - repozitáre

ASICDE

+ Create repository

My repositories | My account | Log out

Hi John,

Here are your repositories

Repository 1 active

The description goes here

Synchronized with git: <https://www.github.com/katrn/repo1.git>

Repository 2 active

The description goes here

Synchronized with git: <https://www.github.com/katrn/repo2.git>

Repository 3 active

The description goes here

Local

Repository 4 active

The description goes here

Local

Repository 5 active

The description goes here

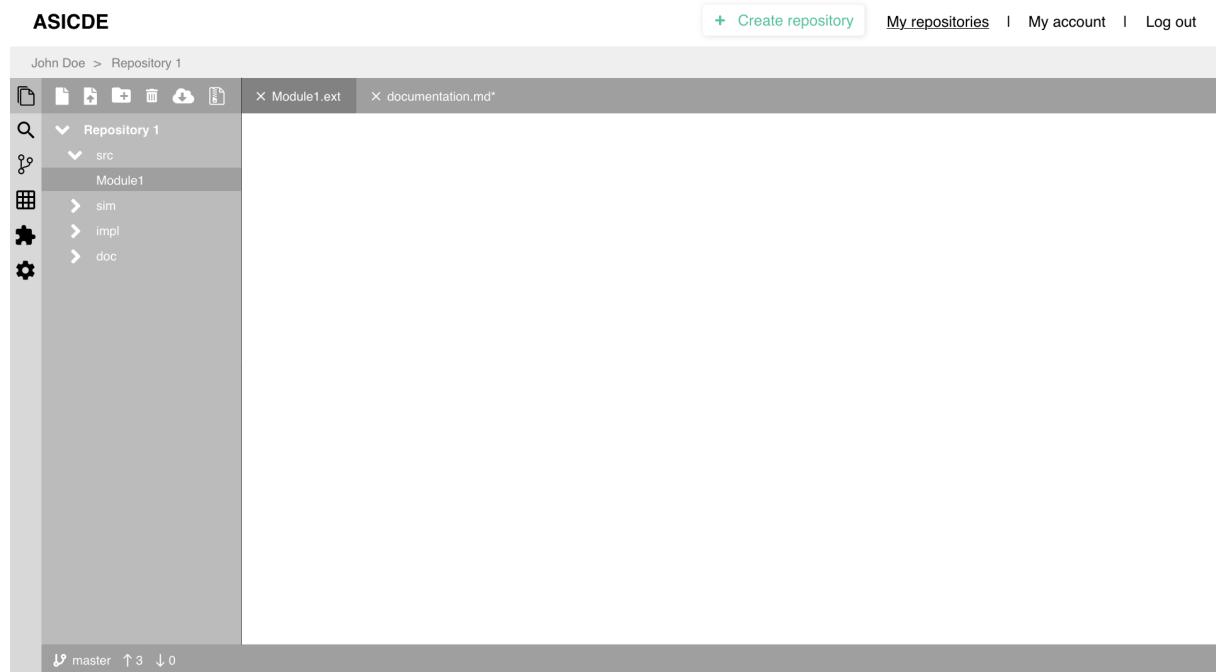
Local

Repository 6 archived

The description goes here

Synchronized with git: <https://www.bitbucket.org/repo6.git>

Editor



Správa členov organizácie

The screenshot shows the ACME organization overview. At the top, it displays 'ASICDE: ACME.org' and navigation links for 'My organization', 'My repositories', 'My account', and 'Log out'. Below this is a section titled 'ACME org overview' with a 'Members' tab selected. Underneath, it says 'Organization members'. A table lists four members:

Name	Role	Teams
Joe Blow j.blow	Owner	3 teams
Steve Blow s.blow	Member	3 teams
Billy Blow b.blow	Member	1 team
Jane Blow j.blowz	Member	4 teams

Správa tímov

ASICDE: ACME org > Team \$Name

[My organization](#) | [My repositories](#) | [My account](#) | [Log out](#)

Team \$Name

Team members

	Tadeáš Drahovský Team Leader		Karolina Trnovecová Member

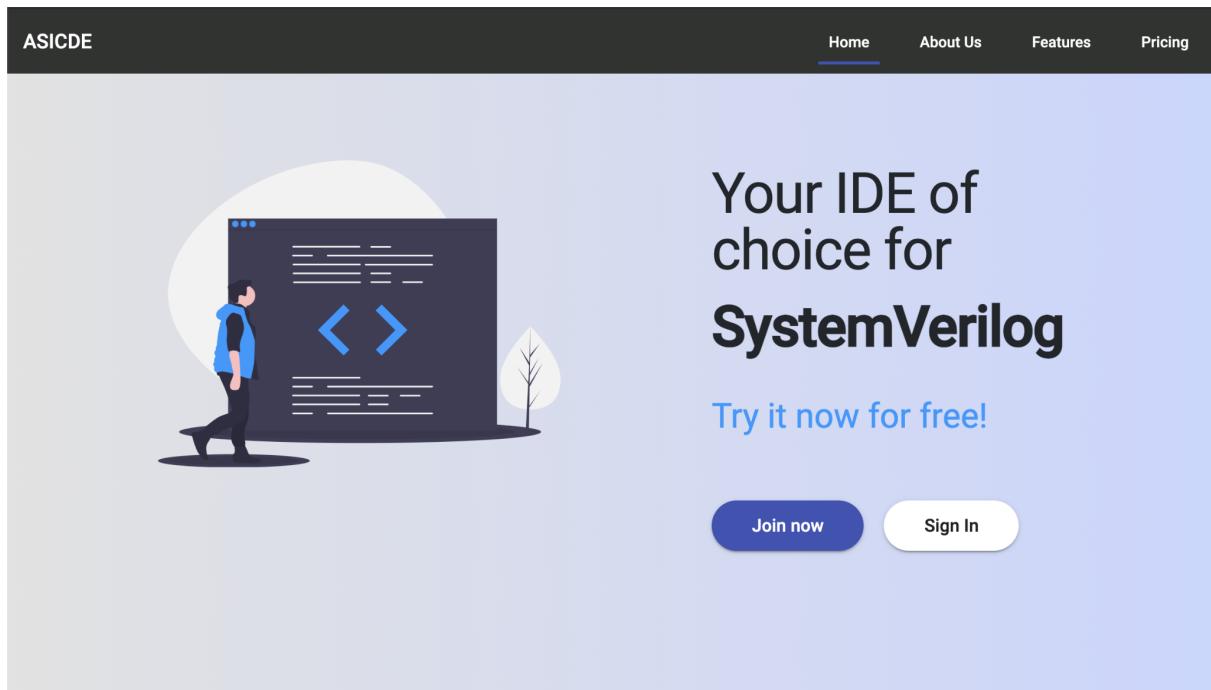
Team repositories

Repository 1 <small>active</small>	The description goes here	
Repository 2 <small>archived</small>	The description goes here	

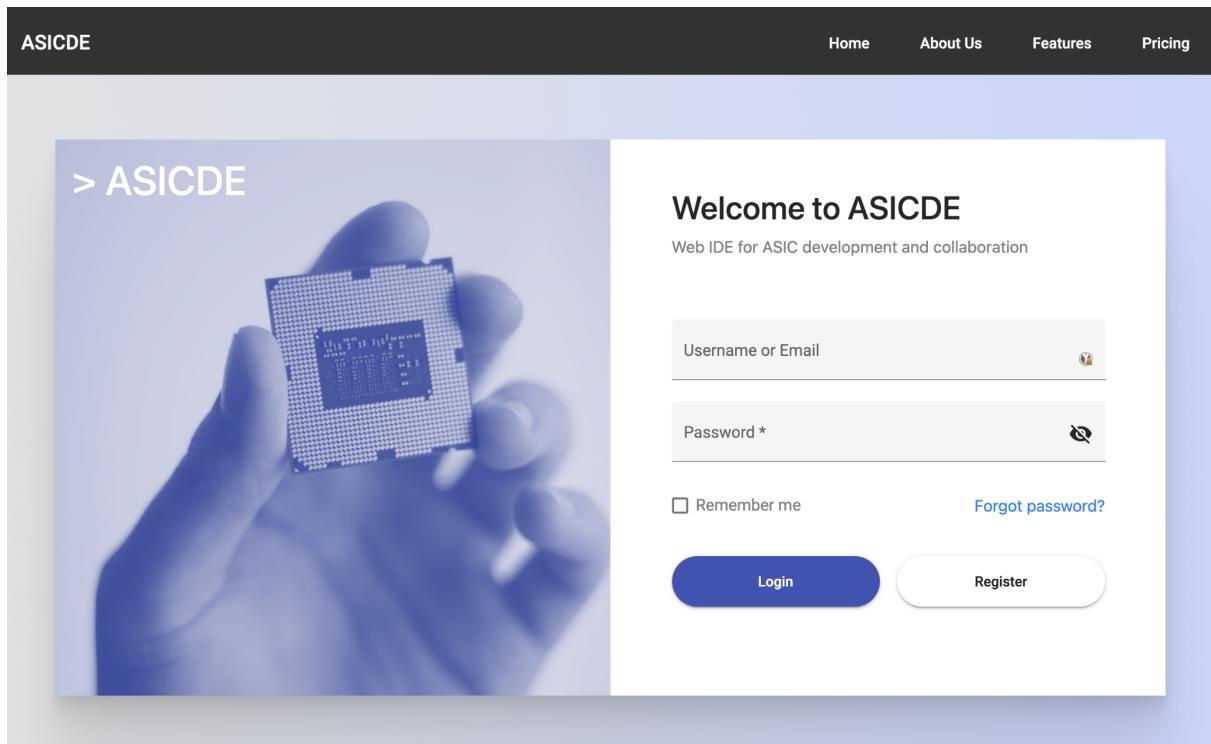
[+ Create a new repository](#)

Príloha C: Frontend screenshoty

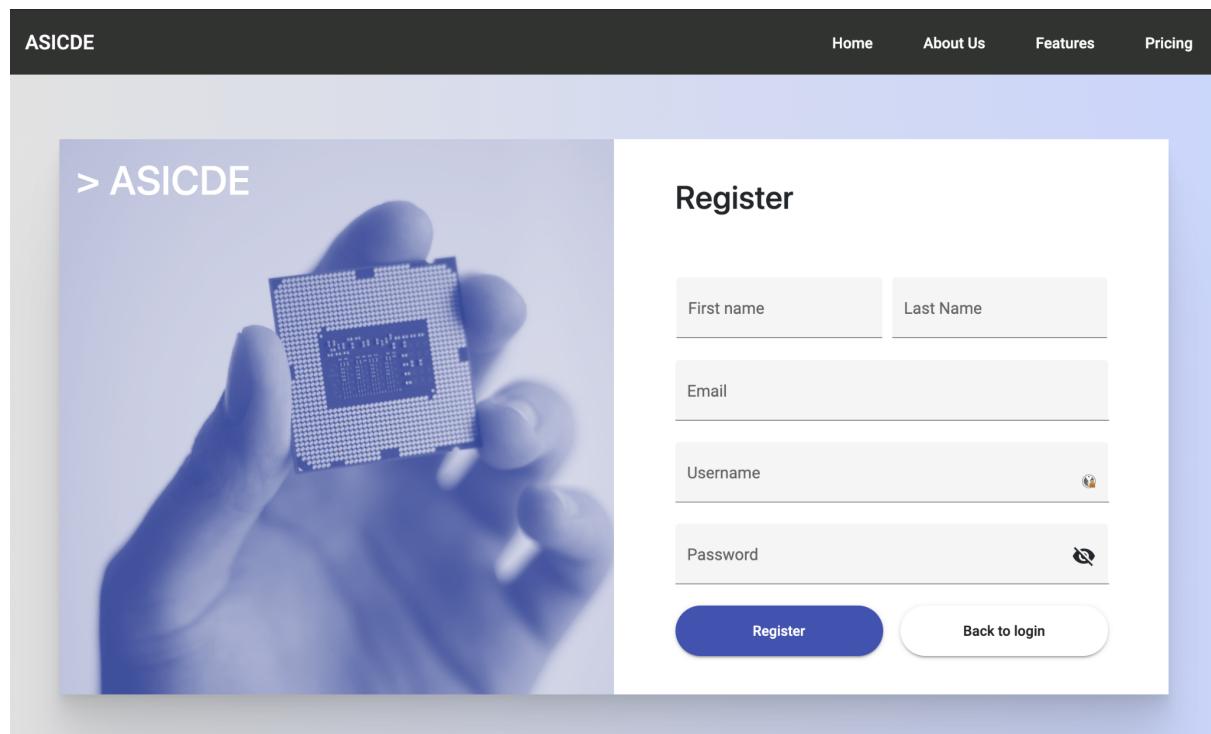
Úvodná stránka



Prihlásenie



Registrácia



The registration page for ASICDE features a large background image of a hand holding a microchip. The text '> ASICDE' is positioned in the top-left corner of the image. To the right of the image is a registration form titled 'Register'. The form includes fields for First name and Last Name, Email, Username, and Password. There are also 'Register' and 'Back to login' buttons.

ASICDE

Home About Us Features Pricing

> ASICDE

Register

First name

Last Name

Email

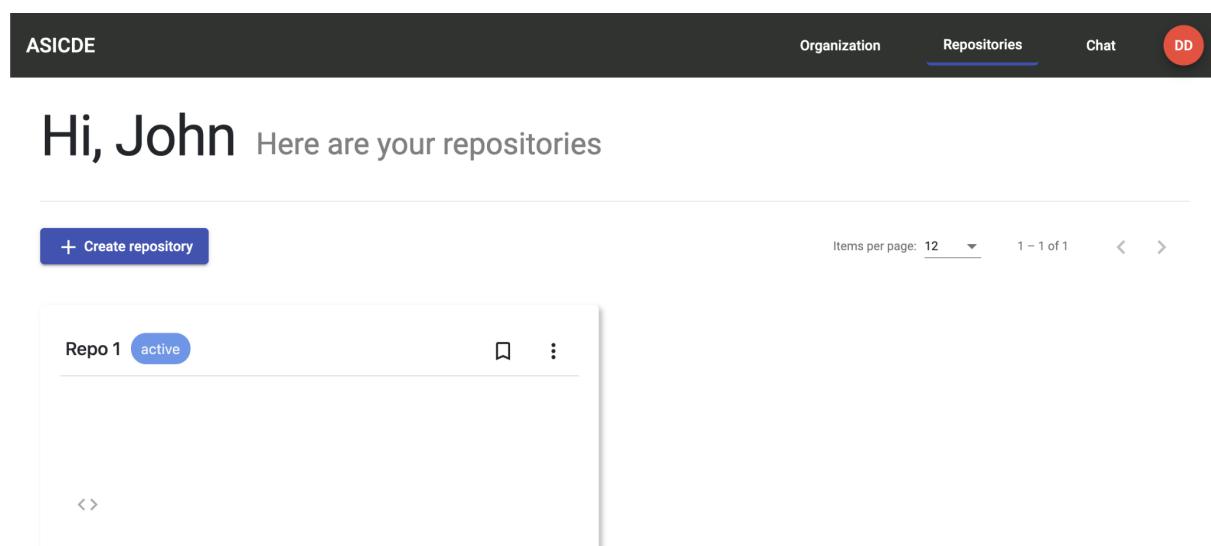
Username

Password

Register

Back to login

Domovská stránka - repozitáre



The repository dashboard for ASICDE shows a greeting 'Hi, John' followed by the message 'Here are your repositories'. It includes a 'Create repository' button and a list of one repository named 'Repo 1' (active). The dashboard also features navigation controls for items per page (12), page number (1 - 1 of 1), and arrows.

ASICDE

Organization Repositories Chat DD

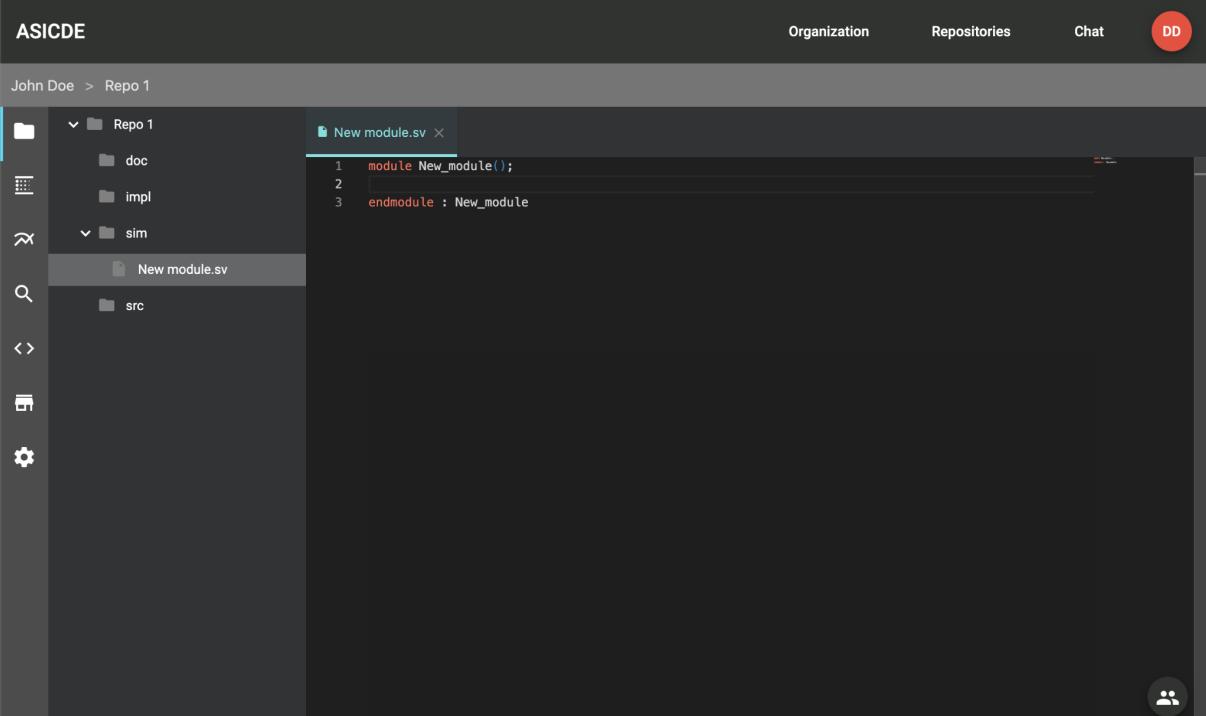
Hi, John Here are your repositories

+ Create repository

Items per page: 12 1 - 1 of 1 < >

Repo 1 active

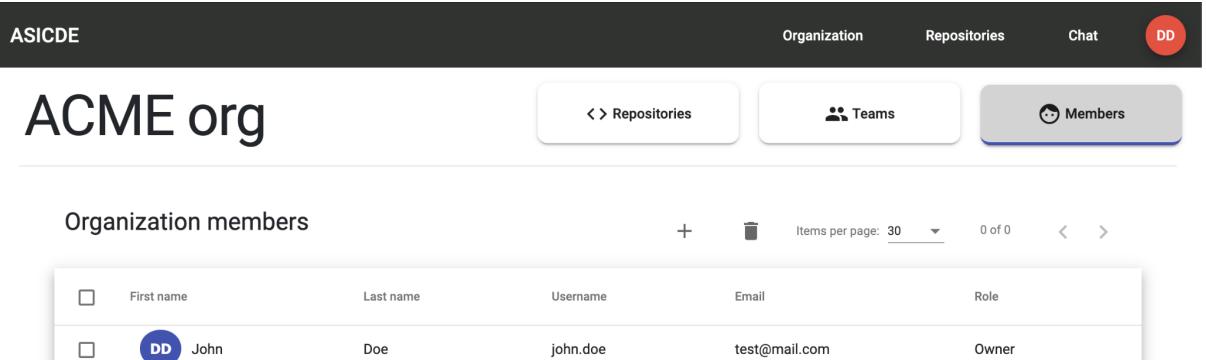
Editor



The screenshot shows the ASICDE Editor interface. At the top, there's a navigation bar with 'ASICDE' on the left and 'Organization', 'Repositories', 'Chat', and a red circular icon with 'DD' on the right. Below the navigation bar, the path 'John Doe > Repo 1' is displayed. On the left, a sidebar shows a file tree for 'Repo 1' with folders 'doc', 'impl', 'sim', and 'src', and a file 'New module.sv' selected. The main area is a code editor with the following content:

```
1 module New_module();
2
3 endmodule : New_module
```

Správa členov organizácie



The screenshot shows the ASICDE Organization members page for 'ACME org'. At the top, there's a navigation bar with 'ASICDE' on the left and 'Organization', 'Repositories', 'Chat', and a red circular icon with 'DD' on the right. Below the navigation bar, there are three tabs: 'Repositories', 'Teams', and 'Members', with 'Members' being the active tab. The main area is titled 'Organization members' and displays a table with the following data:

<input type="checkbox"/>	First name	Last name	Username	Email	Role
<input type="checkbox"/>	DD	John	john.doe	test@mail.com	Owner

Below the table, there are buttons for '+', trash, and search, along with a dropdown for 'Items per page' set to 30, and a status '0 of 0'.

Správa tímov

The screenshot shows the 'ACME org' interface. At the top, there are navigation tabs: Organization, Repositories, Chat, and a red-highlighted Teams tab. Below the tabs, there are three buttons: 'Repositories' (disabled), 'Teams' (selected), and 'Members'.

The left section, titled 'Available teams', shows one item: 'Team coyote' (checked). It includes details: 'Last modified by: john.doe', 'Owned by: John Doe', and a timestamp: 'Sat May 15 2021, 22:58:58 GMT+0200 (Central European Summer Time)'.

The right section, titled 'Team members', shows one item: 'John Doe' (checked). It includes columns: First name, Last name, Username, Email, and Role. The data is: John, Doe, john.doe, test@mail.com, Owner.

Domovská stránka - triedy

The screenshot shows the 'Classrooms' section of the interface. At the top, there are navigation tabs: Classrooms (selected), Repositories, Chat, and a red-highlighted JJ tab. Below the tabs, there are three buttons: 'Create classroom', 'Join classroom', and a dropdown for 'Items per page' set to 12.

The main area displays two classroom entries:

- 1.C**: Slug: 1.C, Created by: [empty]
- 1.D**: Slug: 1.D, Created by: [empty]

Správa triedy

Your classroom X

Students

 Items per page: 10 1 - 1 of 1

 Jan Jansky ⋮

Nickname: jan123
Email: jan.jansky@gmail.com

Assignments

 Items per page: 10 1 - 1 of 1

Cvicensie c.1 ⋮

Created: Tue May 18 2021
Due date: Mon May 31 2021

 To assignment

Príloha D: Špecifikácie modulov projektu

Špecifikácia modulu organizácií

- Tímy môžu mať X repozitárov (napríklad tím pracuje na jednom projekte, prejde na ďalší, tak majú aj starý aj nový - nemusia vytvárať ďalší rovnaký tím, na to aby mali nový repozitár s kódom - takisto napríklad nejaké dev verzie a production verzie kódu? to by mohlo byť v jednom tíme ako dva rôzne repozitáre - ak sa to teda takto rieši)
- Tímy už nemôžu mať žiadne podtímy - keď už je zadefinovaný tím, mal by byť čo najmenší, aby tam boli ľudia len čo majú mať právomoci na všetky repozitáre v tíme, ak je napríklad nový repozitár s kódom ktorému sa venujú ľudia z rôznych tímov, vytvoria nový spoločný tím, pre ten daný projekt a budú tam mať preňho repozitáre
- Môžeme taktiež kontrolovať permissions (read, write, admin) v rámci jednotlivých tímov, teda napríklad aj v rámci tímu by sa keď tak dalo obmedziť že niekto by dostal len read, prípadne by nedostal ani read, a celkovo by obsah tímu (repozitáre) nevidel
- Používateľ ma (alebo je členom) maximálne jednu organizáciu
- Používateľ sa nemôže ani prihlásiť dokým si neoverí emailovú adresu
- Registrácia bude čisto pre bežného používateľa, nebude žiadny výber či chce byť ako individuál alebo organizácia
- Pri prvom prihlásení sa zobrazí tutoriál, že: ahoj, toto je tvoj profil, tuto máš repozitáre, takto vytvoríš nové, tuto máš svoje nastavenia, a ak chceš vytvoriť vlastnú organizáciu, tak klikni sem
- Tutoriál pri prvom zobrazení organizácie (pre ownera) - tu spravuješ používateľov, tu spravuješ tímy, tu repozitáre, a tuto máš ďalšie nastavenia organizácie
- Po založení organizácie sa pozývajú do nej members - pomocou email adresy/username
 - Userovi sa pošle mail notifikácia o tom, že bol pridaný do organizácie, a po prihlásení sa mu zobrazí notifikácia či chce accept/reject pozvanie
 - Ak user nie je v systéme registrovaný, najprv sa registruje, a až potom môže akceptovať invite
- Keď je user už člen organizácie, nemusí byť directly v tíme, figuruje len ako člen org (niečo na štýl MS teams) = môže to byť nejaký manažér, viewer, čo vidí len dashboardy, štatistiky a tak
- Do tímu sa dajú pridať len členovia organizácie, ktorí potvrdili členstvo

- Tímy sa dajú zakladať LEN pod organizáciou, s tým, že keď niekto chce collabovať na jednom projekte, bude sa to riešiť prostredníctvom collabu čo mal na starosti Matúš Pilňan

Špecifikácia edukačného modulu

- môžeš sa zaregistrovať ako učiteľ, kedy budeš vidieť len svoje classrooms (tryedy), a taktiež ich vytvárať (asi ti ostane aj prístup robiť s klasickými repozitárimi, a asi môžeš byť aj v organizácii)
- trieda = skupina študentov, v rámci tryedy učiteľ môže vytvoriť viac zadanií (assignments)
- do tryedy môžeš pridať aj ďalšieho učiteľa, teda ak majú napríklad cviko/predmet spolu..
- assignment čo vytvorí učiteľ je hlavný (parent), keď následne dá publish, tak sa pre každého študenta v tejto classroom vytvorí repo s daným zadáním a priradí sa jemu (owner = student)
- assignment sa potom bude dať znova un-publishnúť, prípadne zmazať, čo zmaže aj všetky ostatné (child)
- taktiež môže zmeniť nastavenia assignmentu, a vidieť a editovať jednotlivé assignemnty - pridať hodnotenie, resp. komentár, atď..
- assignment môže mať ďalšie hodnoty (okrem toho čo je v klasickom repozitári), napríklad čas odovzdania zadania, hodnotenie, komentáre, atď... to sa dá v budúcnosti pridať
- nebola by tam žiadna škola/univerzita, preste keď je niekto zaregistrovaný ako učiteľ, tak by mal možnosť vytvárať classrooms a zadania
- žiak, študent, user, ak je priradený do nejakého classroom-u tak by videl v hornej lište aj My classrooms, okrem My repos alebo My organization, teda mal by prístup aj k education modulu
- študenti si vytvoria ASICDE účty a potom môžu zadať kód (slug) a heslo pre classroom, aby sa do nej dostali
- študenti vidia len svoje assignments, ktoré sú published, a môžu na nich pracovať, ako s normálnym repozitárom
- okrem toho že je to ako normálne repo, budú tam ďalšie informácie naviac - napríklad due_date, komentár učiteľa, hodnotenie.... (možno niekde v nejakej bočnej lište, alebo v kartičke - spomente si na Turing, tam sme mali permanentnú kartičku s dokumentáciou/zadaním - možno niečo také bude fajn)

- učiteľ môže uzavrieť classom - už sa tam nikto iný nemôže prihlásiť
- učiteľ vie meniť nastavenia classroomy, meno, slug, heslo, dávať preč študentov, pozerať si zoznam študentov, vidieť všetky zadania - aj parent aj vypracovávané zadania študentov (asi na to bude nejaké prepínatko, alebo iná kartička, alebo podstránka)

Príloha E: Testovanie

1. ASICDE Testovanie - Posudok

1.1. Hodnotenie používateľského rozhrania

Rozvrhnutie je zrozumiteľné, a dosť jednoduché.

1.2. Hodnotenie funkcionality

Tvorba organizácie je zaujímavá, chat je dosť dobrá myšlienka.

1.3. Hodnotenie komunikácie s tímom (optional)

Super, podľa možností odpoved' hned'.

1.4. Identifikované nedostatky

Jednou vecou čo mi vadí je konzistentnosť, keď je použitý v logine ktorý je veľmi pekný, Angular Material a tiahá sa aplikácia do fialovej farby tak by som v My account nepoužíval ak sa nemýlim Bootstrap ale zjednotil by som formuláre a použil rovnaké farby, taktiež menu s inicialmi používateľa by som nenechal oranžové.

1.5. Zhodnotenie (optional)

Celkovo tímak posunul level tejto aplikácie a vývoj na inú úroveň.

2. ASICDE Testovanie - Posudok

2.1. Hodnotenie používateľského rozhrania

Nové UI je dobre spravené, je prehľadné, zrozumiteľné a intuitívne. V aplikácii sa vďaka nemu dá ľahko orientovať

2.2. Hodnotenie funkcionality

Funkcionalitu ASICDE aplikácie ocenia hlavne vývojári číslicových systémov, ktorí využívajú HDL jazyky. Ja ako softvérový programátor reálne využijem poznatky z programovania tohto prostredia a technológie ktoré som pri tom využíval. Za najdôležitejšiu funkcionalitu, ktorá je v projekte, považujem samotné vytváranie HDL súborov v editore a správu týchto súborov a repozitárov užívateľov.

2.3. Hodnotenie komunikácie s tímom (optional)

Potreboval som pomoc s rozbehnutím projektu na localhoste, ale člen tímu mi bez problémov pomohol so všetkými detailmi a komunikovalo sa veľmi dobre.

2.4. Identifikované nedostatky

Nestrelol som sa so žiadnymi nedostatkami softvéru ani bugmi.