VENTSPILS TEHNIKUMS

EIKT PROJEKTA TEORĒTISKĀ DAĻA

KVALIFIKĀCIJAS DARBS

Autors: Kristaps Kārkliņš

Darba vadītājs: Kārlis Immers

Ventspils 2025

SATURS

IEVADS

Tehnoloģiju attīstība mūsdienās notiek nepārtraukti, laiks kustās uz priekšu, un līdz ar to mainās arī ierīces, risinājumi un pieejamie instrumenti. Katru gadu tiek radītas jaunas metodes un produkti, savukārt iepriekšējās tehnoloģijas noveco un zaudē aktualitāti. Tas nozīmē, ka lietotājiem un speciālistiem arvien biežāk jāpielāgojas jaunumiem, pat ja viņiem trūkst laika vai iespēju sekot līdzi visām pārmaiņām savā jomā.

Tieši šī iemesla dēļ ir būtiski piedāvāt risinājumu, kas palīdz jebkuram interesentam orientēties datoru komponentu pasaulē bez padziļinātām tehniskām zināšanām. Mana izstrādātā mājaslapa ļauj lietotājiem viegli izveidot savu datora komplektāciju, nepārzīstot visas nianses par to, kā komponentes savstarpēji sadarbojas. Sistēma automātiski pārbauda savietojamību un nodrošina, ka lietotājs var izveidot funkcionējošu un uzticamu datoru, balstoties uz saprotamu un intuitīvu pieredzi.

1. UZDEVUMU NOSTĀDNE

Produkta mērķis ir atrisināt sarežģīto un kļūdām pakļauto personālā datora komponentu izvēles procesu, nodrošinot lietotājam iespēju izveidot savietojamu un optimālu datora konfigurāciju bez padziļinātām tehniskām zināšanām. Sistēma samazina risku iegādāties nesavietojamas detaļas.

Lai sasniegtu izvirzīto mērķi, produkts īsteno šādus uzdevumus:

- nodrošināt automātisku komponentu savietojamības pārbaudi (piemēram, procesors–mātesplate, RAM tips u.c.);
- samazināt nepieciešamību pēc ārējas tehniskās konsultācijas;
- nodrošināt pārskatāmu informācijas attēlojumu par izvēlētajiem komponentiem.

Izstrādātais risinājums ir paredzēts plašai lietotāju auditorijai — sākot no pilnīgiem iesācējiem līdz pieredzējušiem speciālistiem. Iesācējiem sistēma nodrošina vienkāršu un drošu datora komplektēšanas procesu bez nepieciešamības pēc padziļinātām tehniskām zināšanām, savukārt pieredzējušiem lietotājiem tā kalpo kā ātrs rīks konfigurāciju pārbaudei un optimizācijai. Platforma ļauj pielāgot komponentus individuālajam budžetam un vajadzībām, nodrošinot ērtu, saprotamu un strukturētu lietošanas pieredzi.

2. PRASĪBU SPECIFIKĀCIJA

2.1. Funkcionālas prasības:

- **Sistēmai automātiski jāveic komponentu savietojamības pārbaude, nodrošinot, ka izvēlētie elementi var darboties kopā.**
- **Lietotājs var sākt konfigurācijas veidošanu ar jebkuru komponenti, un sistēmai automātiski jāpiedāvā tikai tās komponentes, kas ir savietojamas ar jau izvēlētajiem elementiem.**
- **Autorizētiem lietotājiem jābūt iespējai glabāt savas konfigurācijas personīgajā profilā.**
- **Platformai jāpiedāvā iespēja lietotājiem pievienot komentārus gan atsevišķām komponentēm, gan citu lietotāju konfigurācijām.**

2.2 Nefunkcionālas prasības:

- **Programmkoda struktūrai jābūt skaidrai un dokumentētai, lai nodrošinātu efektīvu uzturēšanu, paplašināšanu un komandas sadarbību.**
- **Lietotāja saskarnei jābūt intuitīvai un pieejamai, lai ar sistēmu varētu ērti darboties arī mazāk pieredzējuši lietotāji.**
- **Risinājumam jānodrošina stabila darbība populārākajās pārlūkprogrammās, tostarp Chrome, Firefox un Microsoft Edge.**

3. IZMANTOTĀS TEHNOLOĢIJAS

PHP - servera puses programmēšanas valoda, kas izmantota tīmekļa lietotnes loģikas izstrādei. Plaši izmantota tīmekļa risinājumu izstrādē, tai ir liela izstrādātāju kopiena un plašs bibliotēku klāsts

Laravel - PHP ietvars, kas nodrošina MVC (Model–View–Controller) arhitektūras principu ievērošanu. Tas ļauj strukturēti definēt maršrutus, kontrolierus, modeļus un skatus, kā arī piedāvā iebūvētas drošības, validācijas un testēšanas iespējas.

MySQL- izmanto sistēmas datu glabāšanai un vaicājumu veikšanai. Tā nodrošina strukturētu datu uzglabāšanu, augstu veiktspēju un datu integritāti.

Composer – PHP pakešu pārvaldnieks, kas izmantots ārējo bibliotēku pievienošanai un projekta atkarību pārvaldībai.

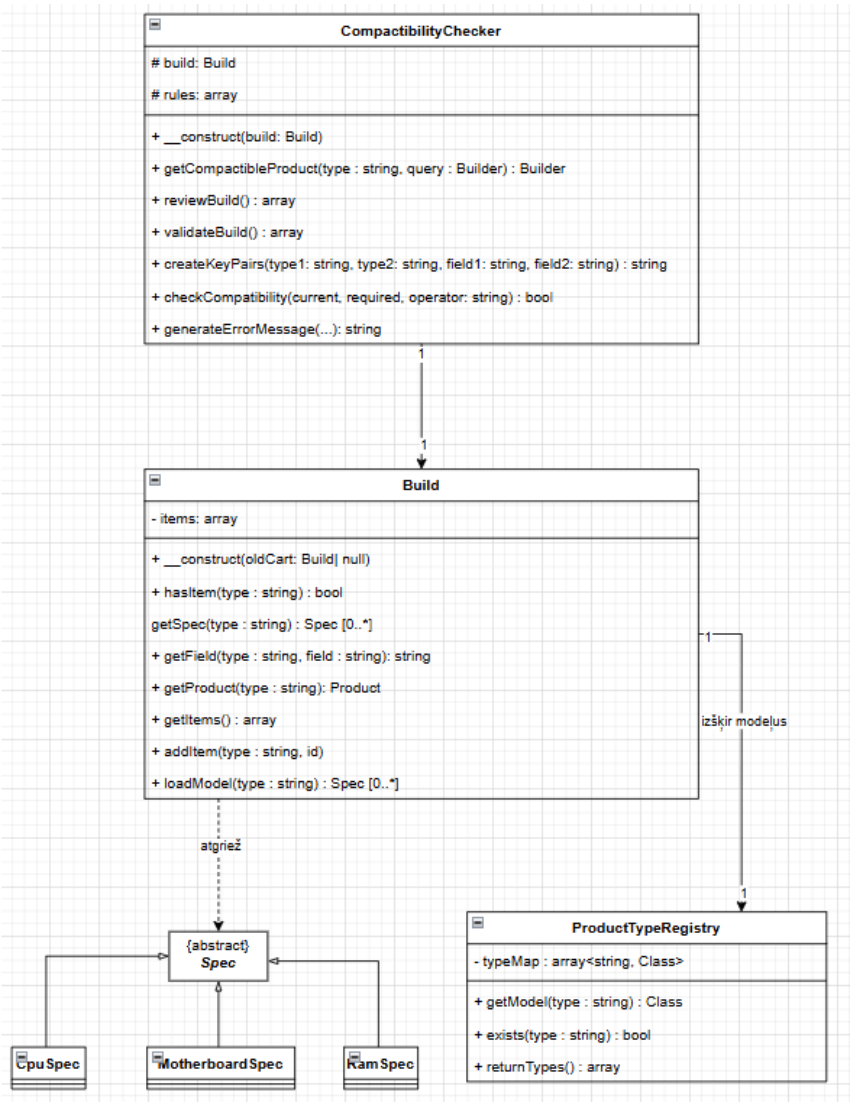
Git - versiju kontroles sistēma, kas nodrošina programmkoda izmaiņu izsekošanu, izstrādes vēstures saglabāšanu un sadarbību starp izstrādātājiem. Tā ļauj droši pārvaldīt koda versijas un atgriezties pie iepriekšējām izmaiņām.

draw.io - vizuālu diagrammu izveides rīks, kas izmantots sistēmas modelēšanai, tostarp blokshēmu, klases diagrammu un datu plūsmas attēlošanai

4. SISTĒMAS UZBŪVE

4.1. Klašu diagramma

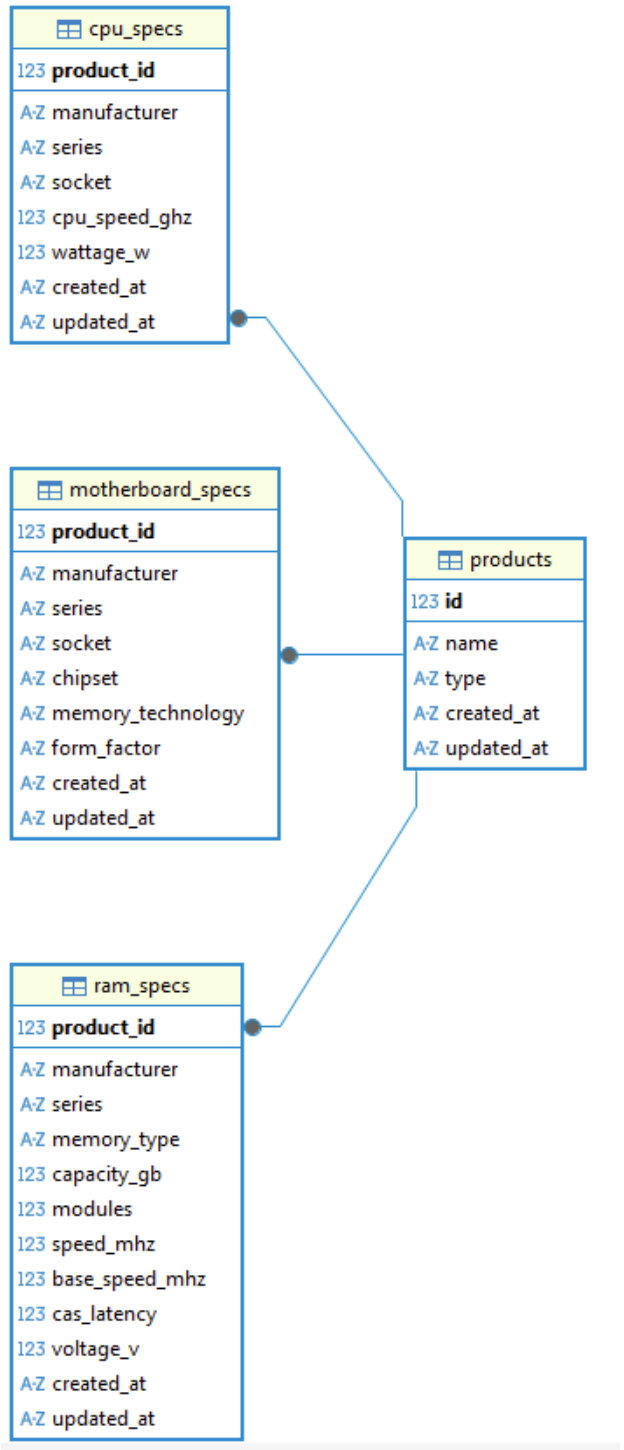
Klašu diagramma attēlo sistēmas galvenās klases, to metodes un savstarpējās attiecības. Build klase uztur lietotāja konfigurāciju, savukārt CompatibilityChecker veic komponentu savietojamības pārbaudi, izmantojot konfigurācijas datus. ProductTypeRegistry nodrošina dinamisku komponentu tipu sasaisti ar atbilstošajiem specifikāciju modeļiem. Specifikāciju klases (CpuSpec, MotherboardSpec, RamSpec) manto kopīgu abstrakto klasi Spec, kas nodrošina vienotu datu apstrādi.



4.1. att. Klašu diagramma

4.2 ER diagramma

ER diagrammā sistēma glabā vispārīgu informāciju par produktiem tabulā products, kur katram ierakstam ir noteikts komponentes tips. Atkarībā no produkta tipa sistēma izveido saistītu ierakstu atbilstošajā specifikāciju tabulā (cpu_specs, motherboard_specs, ram_specs). Specifikāciju tabulas satur tehniskos parametrus, kas tiek izmantoti komponentu savietojamības pārbaudei. Šāda datu struktūra ļauj sistēmai dinamiski atlasīt un salīdzināt komponentes konfigurācijas veidošanas laikā.

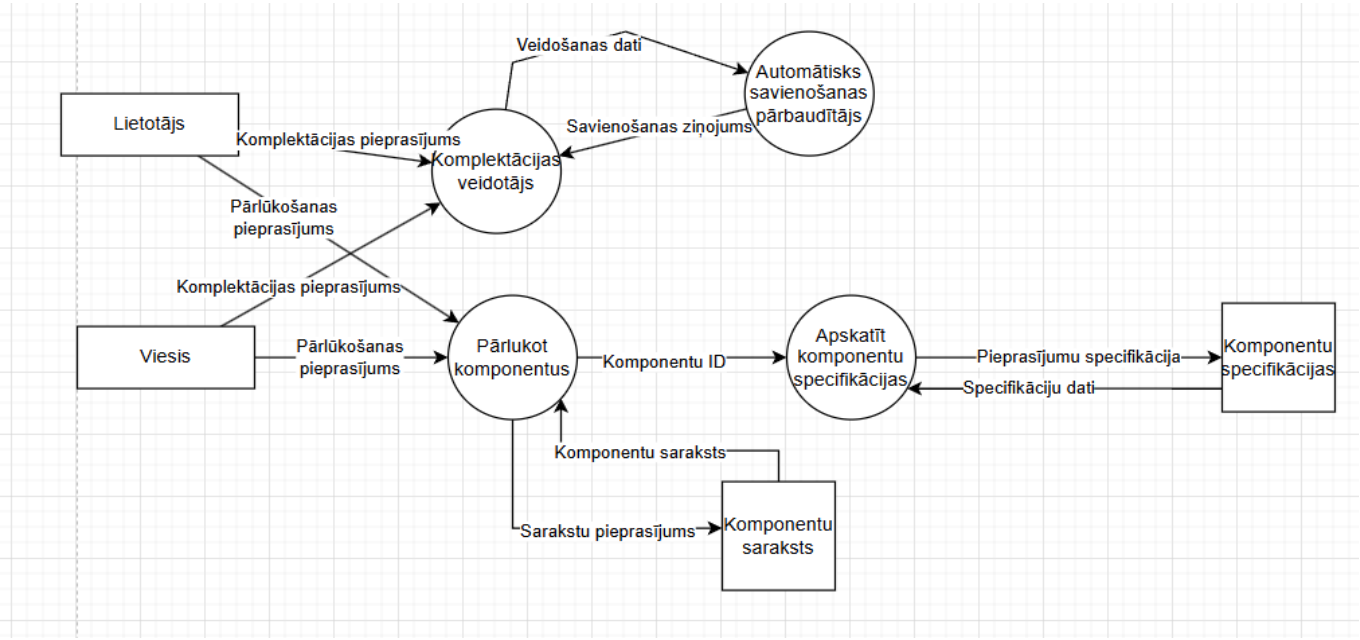


4.2. att. ER diagramma

5. SISTĒMAS DARBĪBAS APRAKSTS

5.1 Datu plūsmu diagramma

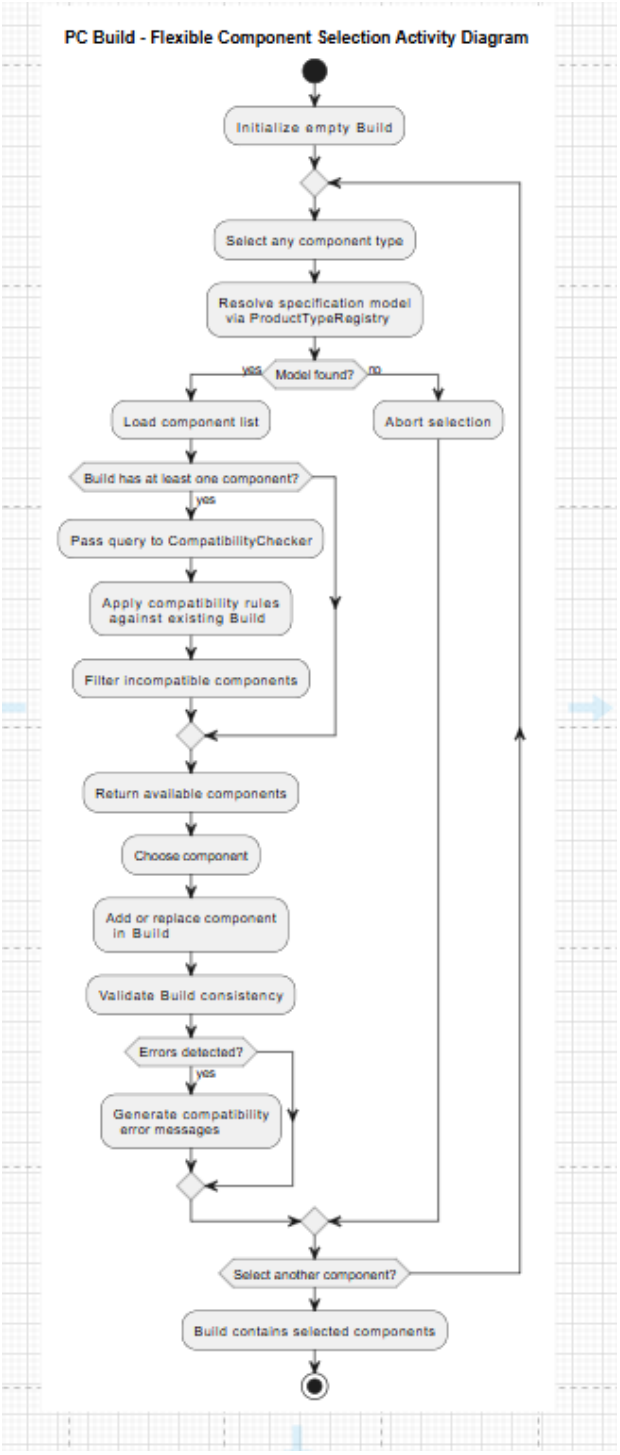
Kā parādīts *5.1. attēlā*, datu plūsmu diagramma attēlo informācijas apriti starp lietotāju, sistēmas procesiem un datu krātuvēm. Diagrammā ir redzams, kā lietotājs pieprasa komponentu sarakstu, kā sistēma apstrādā šo pieprasījumu un kā tiek veikta automātiskā savietojamības pārbaude. Datu plūsmas diagramma palīdz izprast sistēmas darbības loģiku un datu apmaiņu augstākā abstrakcijas līmenī.



5.1. att. Datu plūsmu diagramma

5.2 Aktivitāžu diagramma

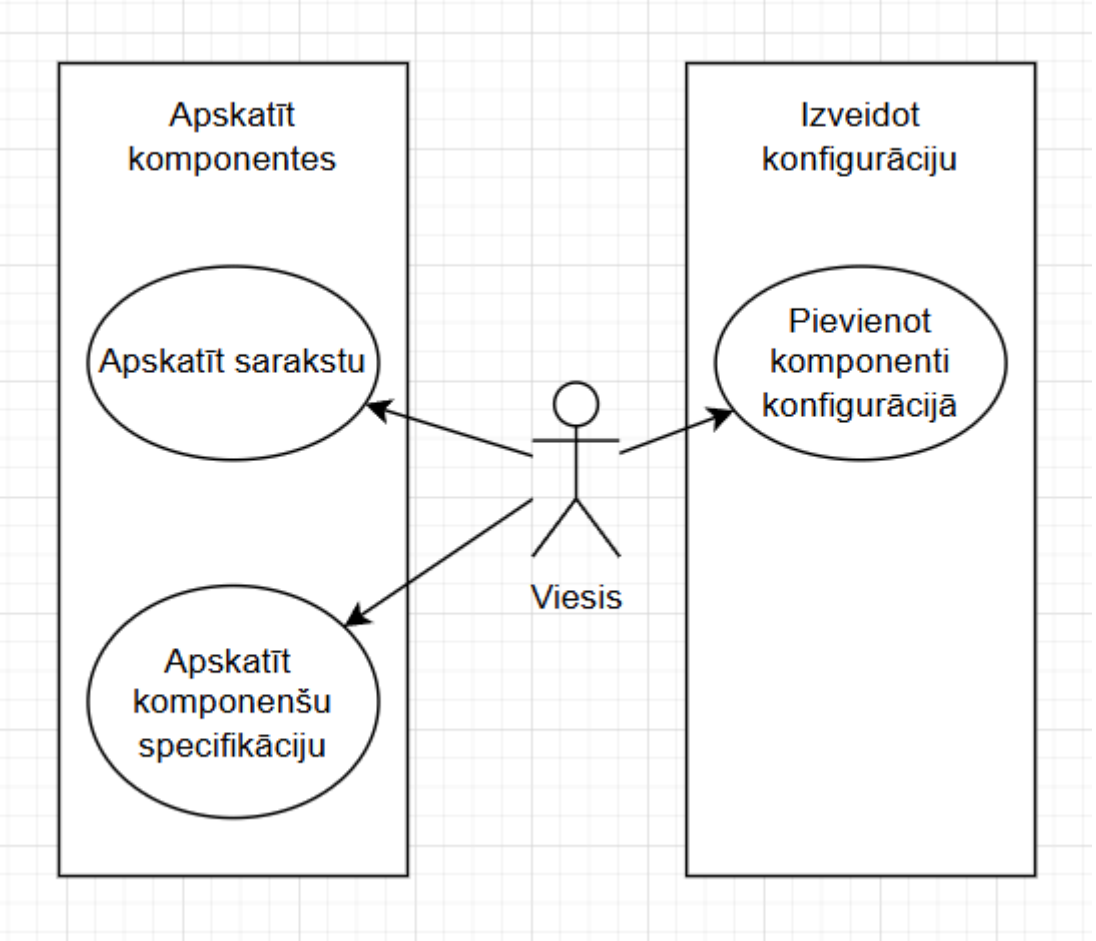
Aktivitāšu diagrammā sistēma inicializē tukšu konfigurāciju un ļauj lietotājam sākt izvēli ar jebkuru komponentes tipu. Pēc komponentes tipa izvēles sistēma ielādē atbilstošo komponentu sarakstu un, ja nepieciešams, filtrē nesaderīgās komponentes. Lietotājs izvēlas komponenti, kas tiek pievienota vai aizvietota konfigurācijā. Pēc katras darbības sistēma pārbauda konfigurācijas korektumu un ģenerē kļūdas, ja tiek konstatēta nesaderība.



5.2. att. Aktivitāšu diagramma

5.3 Lietošanas gadījumu diagramma

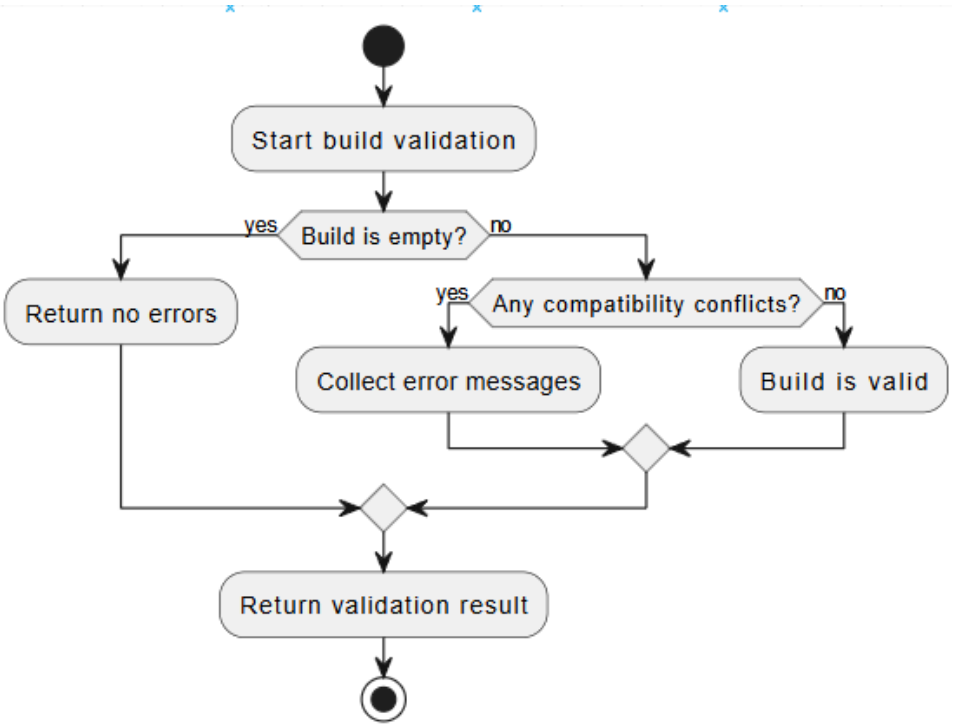
Lietošanas gadījumu diagrammā lietotājs pārlūko komponenšu sarakstus pēc tipa un izvēlas interesējošo komponenti. Sistēma nodrošina detalizētas komponentes informācijas attēlošanu pirms pievienošanas konfigurācijai. Lietotājs var pievienot komponenti konfigurācijai, un sistēma automātiski ņem vērā savietojamības nosacījumus. Visas darbības tiek veiktas bez lietotāja autorizācijas.



5.3. att. Lietošanas gadījumu diagramma

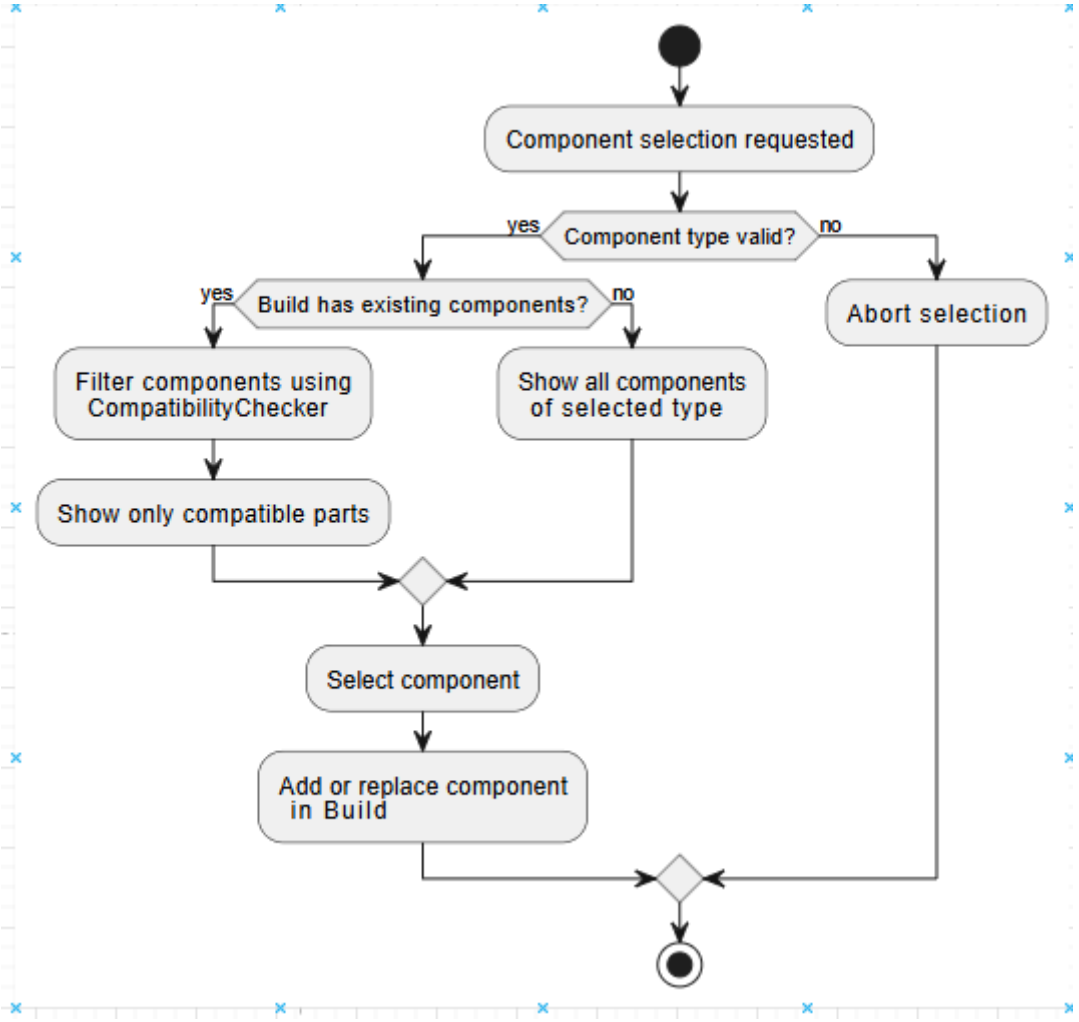
5.4. Blokshēmas

Blokshēmā 5.4.1. sistēma uzsāk lietotāja konfigurācijas validāciju un pārbauda, vai konfigurācija satur vismaz vienu komponenti. Ja konfigurācija ir tukša, validācija tiek pabeigta bez kļūdām. Ja konfigurācijā ir komponentes, sistēma pārbauda to savietojamību savā starpā. Nesaderības gadījumā tiek apkopoti kļūdu paziņojumi un atgriezts validācijas rezultāts.



5.4.1 att. Konfigurācijas validācijas blokshēmas

Blokshēmā 5.4.2. sistēma apstrādā lietotāja komponentes izvēles pieprasījumu. Sākotnēji tiek pārbaudīts, vai izvēlētais komponentes tips ir korekts. Ja tips nav derīgs, izvēles process tiek pārtraukts. Ja tips ir derīgs, sistēma nosaka, vai konfigurācijā jau ir pievienotas citas komponentes. Gadījumā, ja konfigurācija nav tukša, pieejamās komponentes tiek filtrētas, izmantojot savietojamības pārbaudes mehānismu, un lietotājam tiek parādītas tikai savietojamās komponentes. Ja konfigurācija ir tukša, tiek attēlots pilns izvēlētā komponentes tipa saraksts. Pēc komponentes izvēles tā tiek pievienota vai aizvietota konfigurācijā, noslēdzot komponentes izvēles procesu.



5.4.2 att. Komponentes izvēles un filtrēšanas blokshēmas

6. DATU STRUKTŪRAS APRAKSTS

6.1. Specifikāciju tabulas

6.1.1. tabula. Procesoru specifikācijas

Lauks	Tips	Apraksts	Piezīmes
product_id	int	Ieraksta identifikators un saites veidotājs ar tabulu products	Primārā un ārējā atslēga
manufacturer	varchar(255)	Procesora ražotājs	-
series	varchar(255)	Procesoru saime	-
socket	varchar(255)	Procesoru ligzdas tips	Galvenais saderības parametrs
cpu_speed_ghz	decimal(3,1)	Procesora taktsfrekvence GHz	Izsaka gigahercos (GHz)
wattage_w	smallint	Procesora kopējais enerģijas patēriņš vatos (W);	Nedrīkst būt negatīvs, izsaka vatos

6.1.2. tabula. Pamatplašu specifikācijas

Lauks	Tips	Apraksts	Piezīmes
product_id	int	Ieraksta identifikators un saites veidotājs ar tabulu products	Primārā un ārējā atslēga
manufacturer	varchar(255)	Pamatplatnes ražotājs	-
series	varchar(255)	Pamatplatnes saime	-
socket	varchar(255)	Pamatplatnes ligzdas tips	Galvenais saderības parametrs
chipset	varchar(255)	Pamatplatnes mikroshēmu kopne	-
memory_technology	varchar(255)	Pamatplatnes atbalstītais operatīvās atmiņas tips	Galvenais saderības parametrs
form_factor	varchar(255)	Pamatplatnes izmēru standarts	Galvenais saderības parametrs

6.1.3. tabula. Operatīvo atmiņu specifikācijas

Lauks	Tips	Apraksts	Piezīmes
product_id	int	Ieraksta identifikators un saites veidotājs ar tabulu products	Primārā un ārējā atslēga
manufacturer	varchar(255)	Produkta ražotājs	-
series	varchar(255)	Produkta saime	-
memory_technology	varchar(255)	Operatīvās atmiņas tips	Galvenais saderības parametrs
clock_speed	smallint	Atmiņas taktsfrekvence MHz	Galvenais saderības parametrs
ram_size	smallint	Atmiņas apjoms GB	Tikai informatīvs lauks

7. TESTĒŠANA

Prasība Nr. 1: Sistēmai jāspēj ielādēt procesora komponenti un tās specifikāciju, izmantojot komponentes tipu.

Testi:

1. Procesora ielāde, ja konfigurācijā ir pievienots CPU ar korektu ID.
   Rezultāts: sistēma atgriež pamatplates specifikācijas modeli ar piesaistītu produktu.
2. Procesora ielāde, ja konfigurācijā nav pievienots CPU.
   Rezultāts: sistēma atgriež null vērtību.

Prasība Nr. 2: Sistēmai jāspēj ielādēt pamatplates komponenti un tās specifikāciju, izmantojot komponentes tipu.

Testi:

1. Pamatplates ielāde, ja konfigurācijā ir pievienota pamatplate ar korektu ID.
   Rezultāts: sistēma atgriež pamatplates specifikācijas modeli ar piesaistītu produktu.
2. Pamatplates ielāde, ja produkts ar norādīto ID neeksistē datubāzē.
   Rezultāts: sistēma atgriež null.

Prasība Nr. 3: Sistēmai jāspēj noteikt, vai konfigurācijā ir pievienots konkrēts komponentes tips.

Testi:

1. Konfigurācijai tiek pievienota pamatplate, un tiek pārbaudīts tās esamības fakts.
   Rezultāts: metode hasItem() atgriež vērtību true.
2. Tiek pārbaudīts komponentes tips, kas nav pievienots konfigurācijai.
   Rezultāts: metode hasItem() atgriež vērtību false.

Prasība Nr. 4: Sistēmai jāspēj iegūt komponentes specifikāciju, ja konfigurācijā ir pievienots korekts komponentes tips.

Testi:

1. Konfigurācijai tiek pievienots CPU, un tiek izsaukta specifikācijas iegūšanas metode.
   Rezultāts: metode getSpec() atgriež procesora specifikācijas objektu.
2. Tiek mēģināts iegūt CPU specifikāciju, ja konfigurācijā pievienota cita komponente.
   Rezultāts: metode getSpec() atgriež null.

Prasība Nr. 5:Sistēmai jānodrošina korekta darbība, ja tiek pieprasīts neeksistējošs produkts.

Testi:

1. Konfigurācijai tiek pievienots produkts ar ID, kas neeksistē datubāzē.
   Rezultāts: metode loadModel() atgriež null, un sistēma netiek pārtraukta.

Prasība Nr. 6: Sistēmai jāspēj pārrakstīt komponenti, ja viens un tas pats tips tiek pievienots atkārtoti.

Testi:

1. Konfigurācijai tiek pievienoti divi CPU pēc kārtas ar dažādiem ID.
   Rezultāts: konfigurācijā tiek saglabāts tikai pēdējais pievienotais CPU.

Prasība Nr. 7: Sistēmai jāspēj iegūt konkrēta lauka vērtību no komponentes specifikācijas.

Testi:

1. Konfigurācijai tiek pievienots CPU ar definētu socket lauku.
   Rezultāts: metode getField() atgriež ligzdas vērtību.
2. Tiek mēģināts iegūt lauka vērtību, ja komponente nav pievienota konfigurācijai.
   Rezultāts: metode atgriež null.

Prasība Nr. 8: Sistēmai jāspēj atlasīt tikai tās komponentes, kas ir savietojamas ar lietotāja konfigurācijā jau izvēlētajām komponentēm.

1. Konfigurācijai tiek pievienots procesors, un tiek pieprasīts pamatplašu saraksts.
   Rezultāts: sistēma ģenerē datubāzes vaicājumu ar nosacījumu, kas filtrē pamatplates pēc ligzdas (socket) parametra.

1. Tiek pārbaudīts vaicājuma parametru skaits.
   Rezultāts: vaicājumā tiek izmantots viens piesaistītais parametrs, kas atbilst procesora ligzdai.

Prasība Nr. 9: Sistēmai jāidentificē nesaderīgas komponentes lietotāja konfigurācijā un jāatgriež kļūdas paziņojums.

Testi:

1. Konfigurācijai tiek pievienots procesors ar ligzdu LGA1700 un pamatplate ar ligzdu AM5..

   Rezultāts: sistēma atgriež kļūdu masīvu, kas satur ierakstu par nesaderīgo komponentu pāri.

1. Tiek pārbaudīta kļūdas identifikācija pēc komponenšu pāra atslēgas.

   Rezultāts: kļūdu masīvā eksistē atslēga, kas identificē CPU un pamatplates ligzdas nesaderību.

Prasība Nr. 10: Sistēmai jāatļauj konfigurācijas validācija bez kļūdām, ja izvēlētās komponentes ir savietojamas.

Testi:

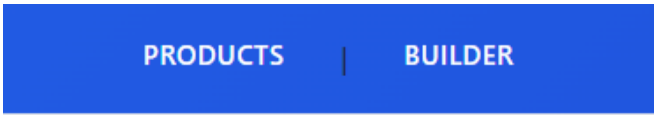
1. Konfigurācijai tiek pievienots procesors un pamatplate ar vienādu ligzdas tipu (LGA1700).

   Rezultāts: sistēma neveido kļūdu paziņojumus un atgriež tukšu kļūdu masīvu.

8. LIETOTĀJU CEĻVEDIS

8.1 Navigācijas josla

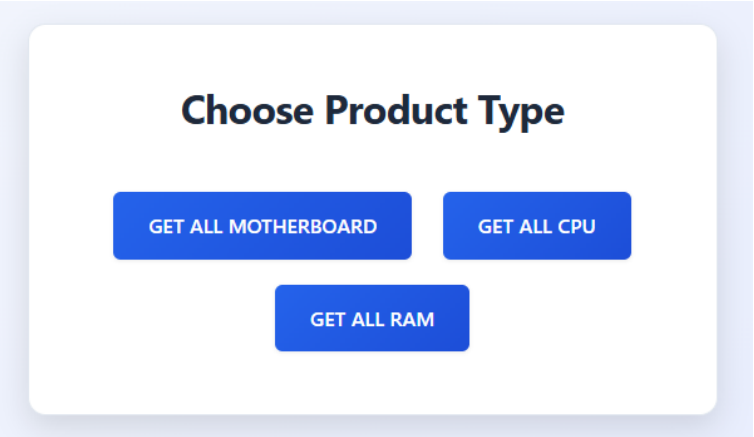
Kā redzams *8.1. attēlā*, sistēmas navigācijas josla nodrošina galveno pārvietošanos starp sistēmas sadaļām. Tajā ir pieejamas saites uz komponenšu tipu apskati, konfigurācijas veidošanas sadaļu. Navigācijas josla ir pastāvīgi redzama, kas uzlabo lietojamību un ļauj lietotājam ātri piekļūt nepieciešamajām darbībām neatkarīgi no atrašanās vietas sistēmā.



8.1. att. Navigācijas josla

8.2 Komponenšu tipu saraksts

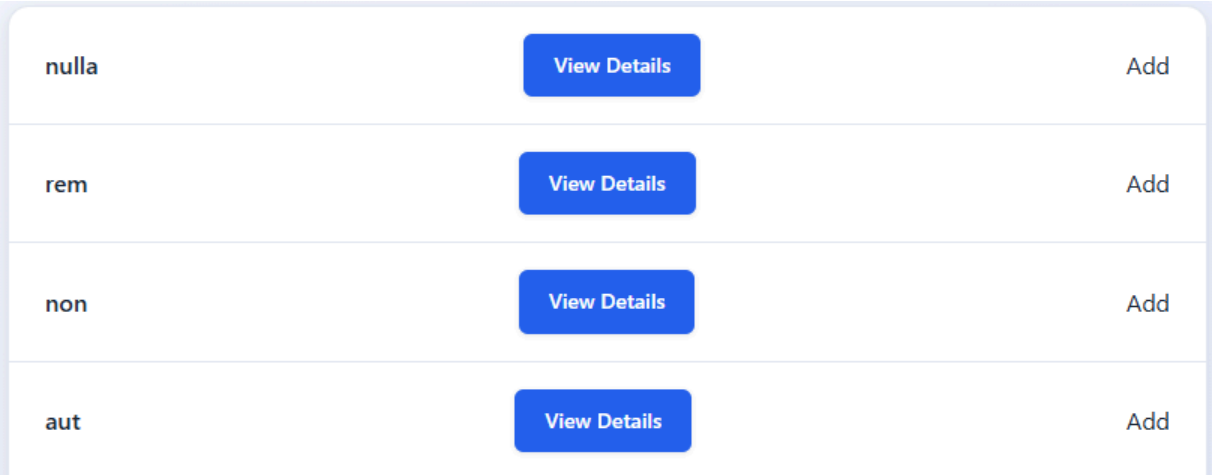
8.2. attēlā ir attēlots piedāvāto komponenšu tipu saraksts. Lietotājam tiek piedāvāta iespēja izvēlēties konkrētu komponentes kategoriju, piemēram, procesorus, pamatplates vai operatīvo atmiņu. Šī funkcionalitāte ļauj strukturēti pārlūkot sistēmā pieejamās komponentes un kalpo kā sākuma punkts konfigurācijas veidošanai.



8.2. att. Komponenšu tipu saraksts

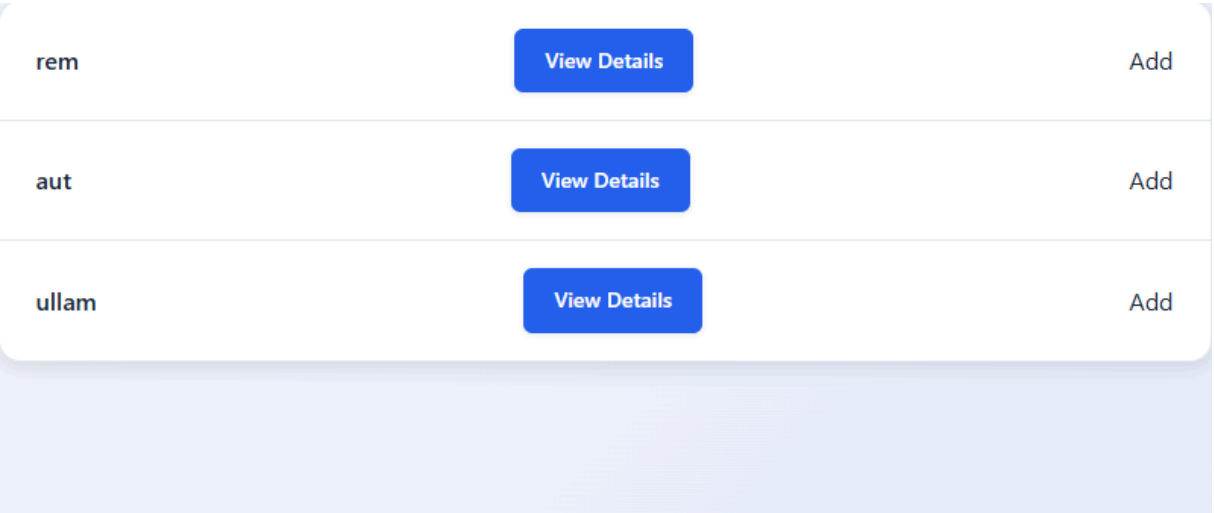
8.3 Izvēlēto komponenšu saraksts

Kā redzams *8.3.1. attēlā*, lietotājam tiek parādīts pieejamo komponenšu saraksts izvēlētajā kategorijā. Katrā ierakstā ir norādīts komponentes nosaukums un pieejamas darbības, piemēram, detalizētas informācijas apskate vai komponentes pievienošana konfigurācijai. Šāds izkārtojums nodrošina skaidru un pārskatāmu izvēles procesu.



8.3.1. att. Komponenšu izvēles saraksts

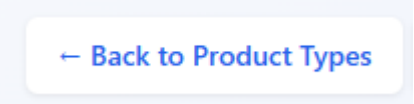
Kā redzams *8.3.2. attēlā*, sistēma attēlo komponenšu sarakstu, kurā tiek parādītas tikai tās komponentes, kas ir savietojamas ar lietotāja jau izveidoto konfigurāciju. Komponenšu filtrēšana notiek automātiski, ņemot vērā iepriekš izvēlētos komponentu parametrus, piemēram, ligzdas tipu, atmiņas tehnoloģiju un citus saderības kritērijus. Šāda pieeja novērš nesaderīgu komponenšu izvēli un atvieglo lietotājam pareizas konfigurācijas izveidi.



8.3.2. att. Savietojamo komponenšu saraksts

8.4 Atgriešanās poga uz komponenšu tipiem

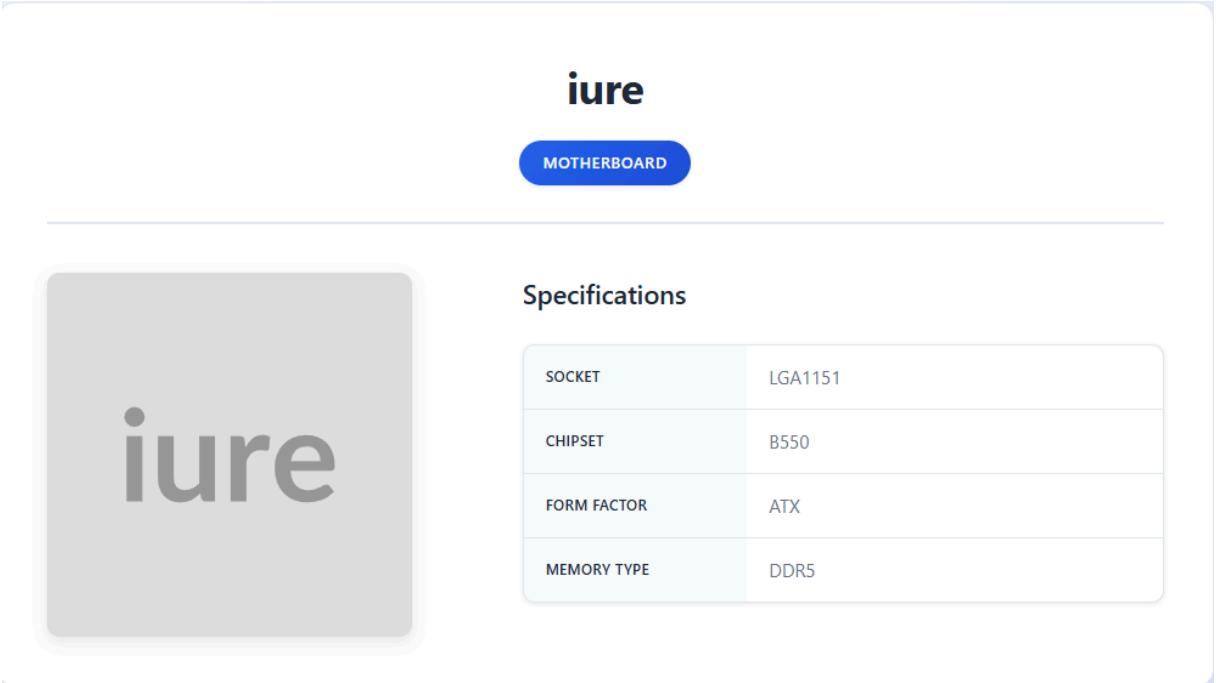
Kā parādīts *8.4. attēlā*, sistēmā ir pieejama poga, kas ļauj lietotājam atgriezties pie komponenšu tipu saraksta. Šī funkcija nodrošina ērtu navigāciju un ļauj lietotājam mainīt izvēli bez nepieciešamības atkārtoti atvērt sākuma lapu.



8.4. att. Atgriešanās poga uz komponenšu tipiem

8.5 Komponentes detalizētais apraksts

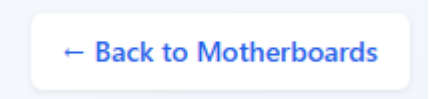
8.5. attēlā ir attēlots izvēlētās komponentes detalizētais skats. Tajā lietotājs var apskatīt komponentes attēlu un galvenos tehniskos parametrus, piemēram, ligzdas tipu, mikroshēmu kopni, formfaktoru un atbalstīto atmiņas tipu. Šī sadaļa sniedz lietotājam pilnīgu informāciju, kas nepieciešama, lai pieņemtu pamatotu lēmumu par komponentes izvēli.



8.5. att. Komponentes detalizētais apraksts

8.6 Atgriešanās poga uz sarakstu

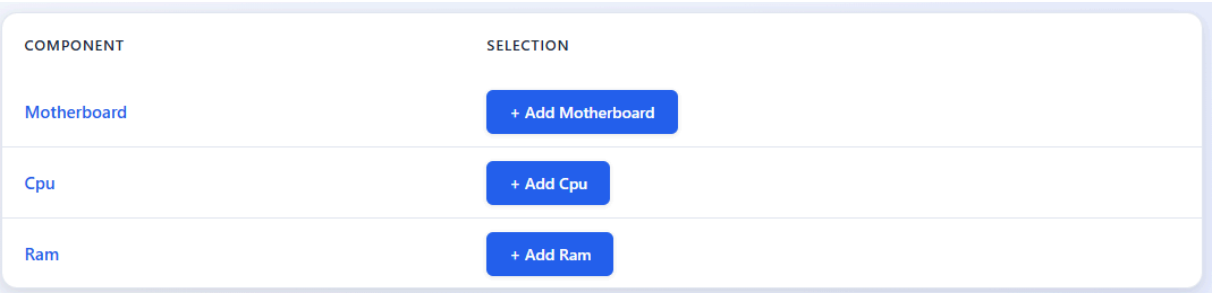
Kā redzams *8.6. attēlā*, lietotājam ir iespēja atgriezties pie saraksta pēc komponentes detalizētas informācijas apskates. Šī funkcionalitāte uzlabo lietošanas pieredzi, ļaujot ātri pāriet starp dažādiem komponentu aprakstiem.



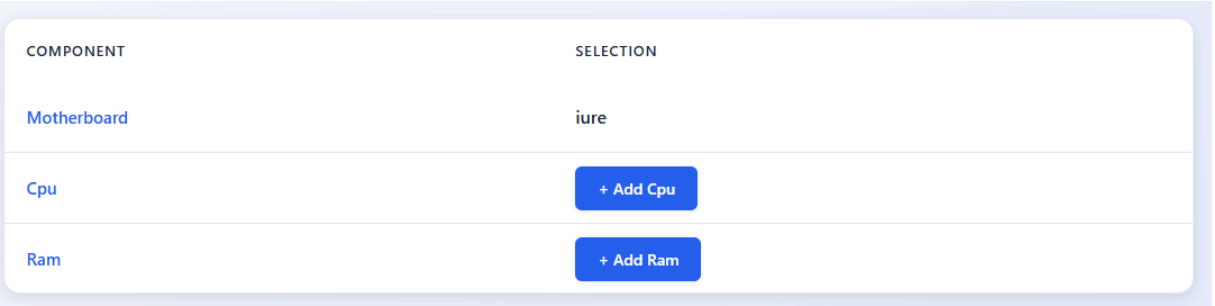
8.6. att. Atgriešanās poga uz sarakstu

8.7 Konfigurācijas komponenšu pārvaldība

8.7. attēlā ir attēlota konfigurācijas pārvaldības sadaļa, kurā lietotājs var pievienot dažādas komponentes, piemēram, pamatplati, procesoru vai operatīvo atmiņu. Šī sadaļa nodrošina centralizētu konfigurācijas veidošanu un ļauj lietotājam pakāpeniski komplektēt datoru, vienlaikus izmantojot sistēmas savietojamības pārbaudes mehānismu.



8.7.1. att. Konfigurācijas komponenšu pārvaldība



8.7.2. att. Konfigurācijas piemērs ar pievienotu komponenti

SECINĀJUMI

Darba gaitā tiek veidota sistēmas loģika, kas balstās uz dinamisku pieeju programmēšanā. Šīs pieejas praktiskā realizācija ir parādīta 1. pielikumā, kurā attēlots risinājums komponentu tipu sasaistīšanai ar atbilstošajiem datu modeļiem.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] [Tiešsaistē]. Pieejams:

PROGRAMMAS PIRMKODS

GitHub: RudisLV2006

PIELIKUMI

1. Pielikums

Attēlots klases realizācijas piemērs, kas nodrošina komponentu tipu sasaisti ar atbilstošajiem specifikāciju modeļiem. Šī klase ir nozīmīga sistēmas arhitektūras sastāvdaļa, jo ļauj dinamiski noteikt, kāda tipa dati jāielādē atkarībā no lietotāja izvēlētās komponentes.

```php
class ProductTypeRegistry
{
    protected static array $typeMap = [
        'motherboard' => MotherboardSpec::class,
        'cpu'  => CpuSpec::class,
        'ram' => RamSpec::class,
    ];
    public static function getModel(string $type): ?string
    {
        return self::$typeMap[$type];
    }
    public static function exists(string $type): bool
    {
        return isset(self::$typeMap[$type]);
    }
    public static function returnTypes()
    {
        return array_keys(self::$typeMap);
    }
}
```