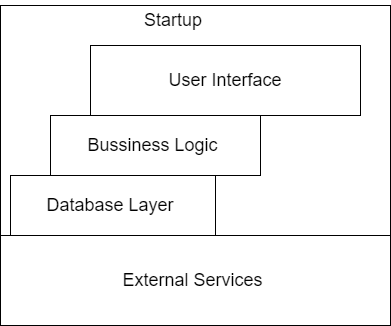
|  |  |
| --- | --- |
| **Layer** | **Responsibilities** |
| Startup | Lidhja e sistemit, leximi, menaxhimi i sesioneve të përdorueseve. |
| User Interface | Implementimi i Userinterface për përdoruesit si tek pagesat, tek shtimi i punëtorëve, performaca etj. |
| Business Logic | Përpunimi i kërkesave të pi kërkesave të përdoruesve si dhe validimi i të dhënave. |
| Database Layer | Ruajtja e të dhënave, rikthimi i të dhënave oraret etj. |
| External Layer | Mundësia e komunikimit me API të jashtëm për sistemimin e pagesave, pushimeve. |



2.2 Startup Layer Decomposition

Tek sistemi për menaxhimin e punëtorëve klasa Startaup është klasa kryesore e programit, ky program është i shkruar në Python. Shtresa Startup ka për detyrë konfigurimin dhe inicimin e komponenteve të sistemit, bazuar në specifikimet e dhëna në kohën e tyre.

|  |  |
| --- | --- |
| Module | Responsibilities |
| Employee System | Konfigurimi i komponenteve kryesore, dhe pergjegjesia e gjithë sistemit. |
| Function of Employee | Create, Read, Update dhe Delete |
| Configurator | Leximi dhe ngarkimi i konfigurimeve |

2.2.2 Startup Layer Interface Specifications

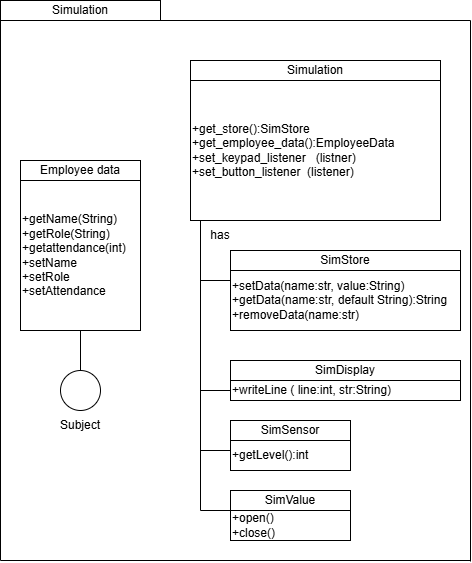
|  |  |  |
| --- | --- | --- |
| Python initialization | Syntax: |  |
| Pre: |  |
| Post: |  |
| Program execution | Syntax: |  |
| Pre: |  |
| Post: |  |

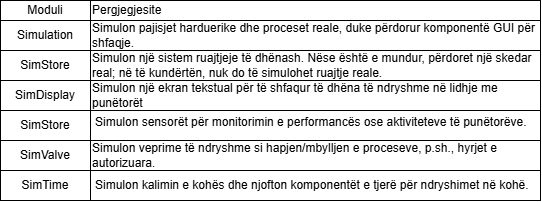
**Table B-10-5 Startup Layer Interface Specifications**

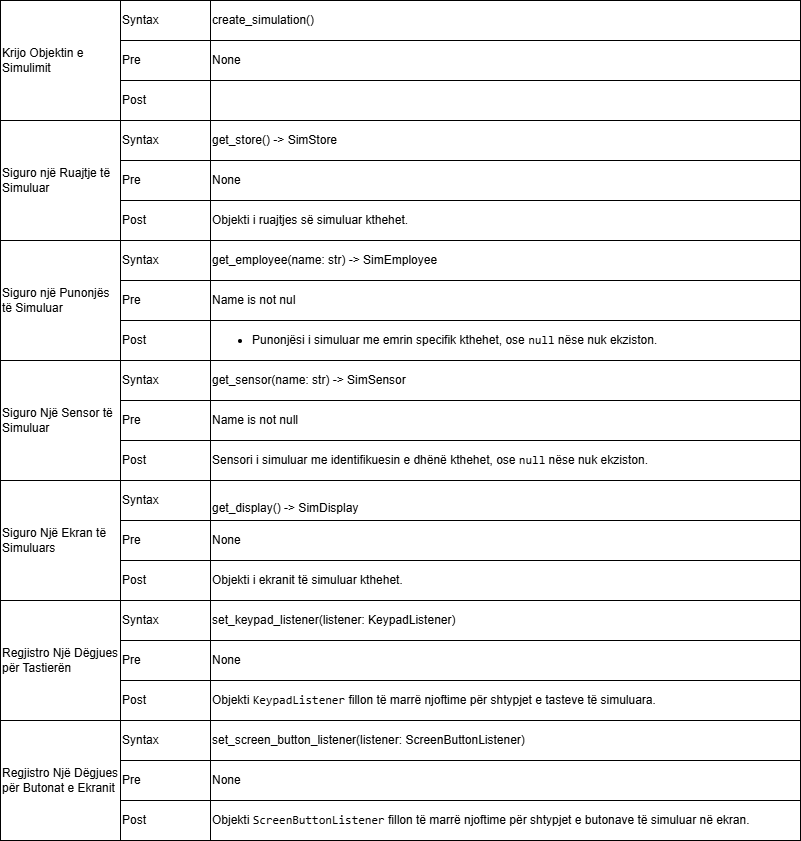
Services Required

2.2.3 Startup Layer Desing Rationale

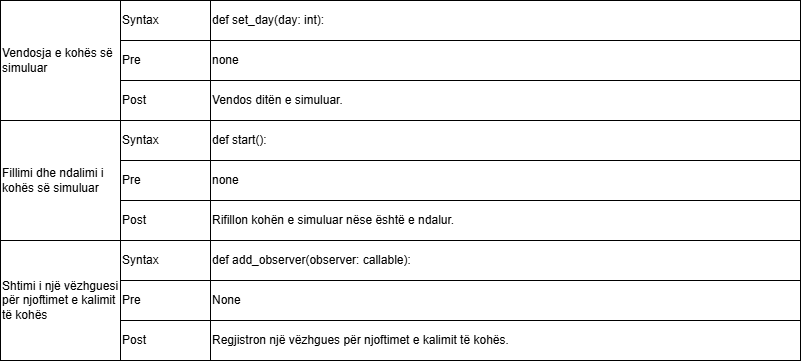
Detyra e konfigurimit mund të ishte vendosur në klasat kryesore të aplikacionit, por kjo do të çonte në kod të tepërt. Duke e ndarë këtë funksionalitet në një komponent të dedikuar, si **Configurer**, ai bëhet i ripërdorshëm, më i lehtë për t'u mirëmbajtur dhe më i thjeshtë për t'u modifikuar. Edhe pse projektimi fillestar kërkon më shumë përpjekje, në planin afatgjatë kursen kohë.  
Ky komponent **Configurer** do të implementojë modelin **Abstract Factory**, i cili mundëson integrimin pa probleme të komponentëve të ndryshëm virtualë (si module për regjistrat e punonjësve, sistemet e pagave ose ndjekjen e performancës)











2.2.4 Startup Layer Desing Rationale

Shtresa e Simulimit imiton mjedisin e EMS dhe përfshin komponentë kryesorë si: Simulatorin e Kohës, që menaxhon oraret dhe pjesëmarrjen; Simulatorin e Ruajtjes së Përhershme, që trajton ruajtjen e të dhënave të punonjësve; Simulatorin e Pajisjes së Shfaqjes, që ofron një ndërfaqe vizuale për raporte dhe të dhëna; dhe Simulatorë të Sensorëve dhe Veprimeve, që imitojnë integrime si biometrika ose njoftimet. Një klasë për thjeshtimin e aksesit te komponentëve të tjerë, si një panel administrativ i simuluar, ndërveprimet me kioskat dhe mjedisin e vendit të punës.

2.3.3 Simulation Layer Design Rationale

U morën në konsideratë disa alternativa për shtresën e Simulimit, por ato mungonin në disa komponentë ose kishin funksionalitete të vendosura në shtresa të tjera. Për shembull, ruajtja e përhershme nuk ishte fillimisht në këtë shtresë, derisa u kuptua se një applet mund të mos lejohet të ketë akses në sistemin operativ pritës për shkak të çështjeve të sigurisë. Në fund, u kuptua se kjo shtresë duhet të simulojë gjithçka në mjedisin e softuerit.

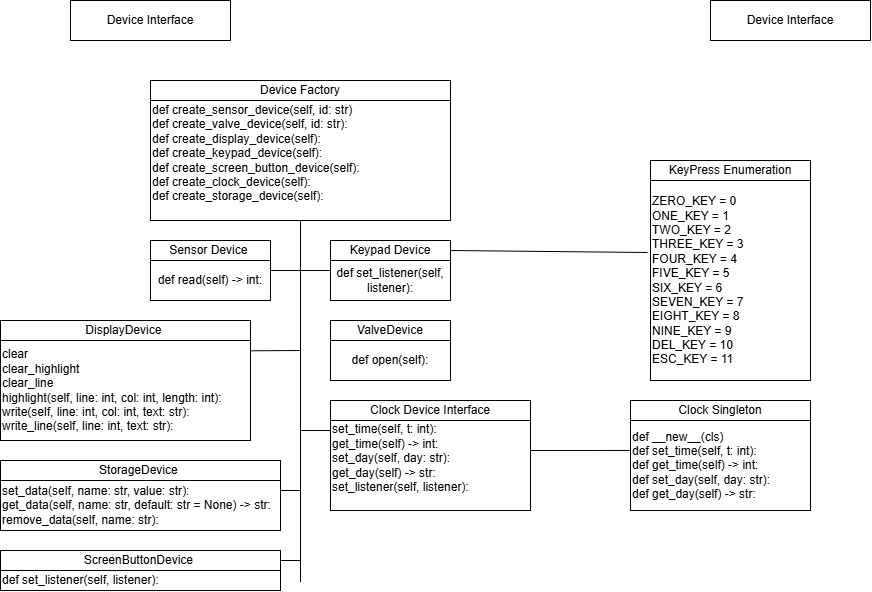
Si shembull tjetër, në një alternativë, procesi i ndërtimit të GUI-së së simulimit u la pjesërisht në dorë të applet-it, duke kërkuar një panel kontrolli dhe një mjedis të simuluar nga shtresa e Simulimit dhe duke i kombinuar ato për të formuar applet-in. Alternativa aktuale është më kohezive.

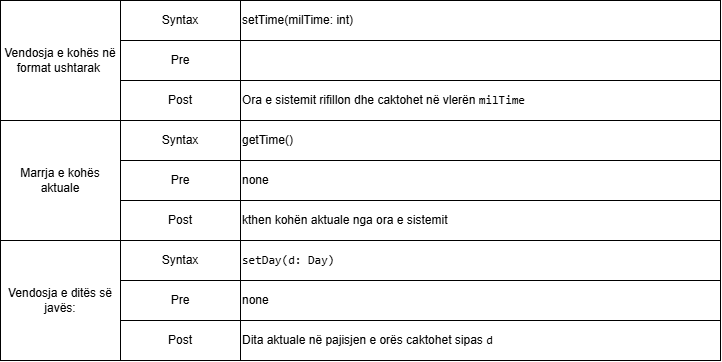
Klasa **SimTime** është një klasë **singleton**. Kjo u konsiderua e nevojshme sepse pasja e më shumë se një objekti të kohës së simuluar do të shkaktonte me siguri dështimin e programit. Pasi **SimTime** bëhet singleton, nuk ka nevojë për një operacion për ta tërhequr nga fasada e shtresës së Simulimit.

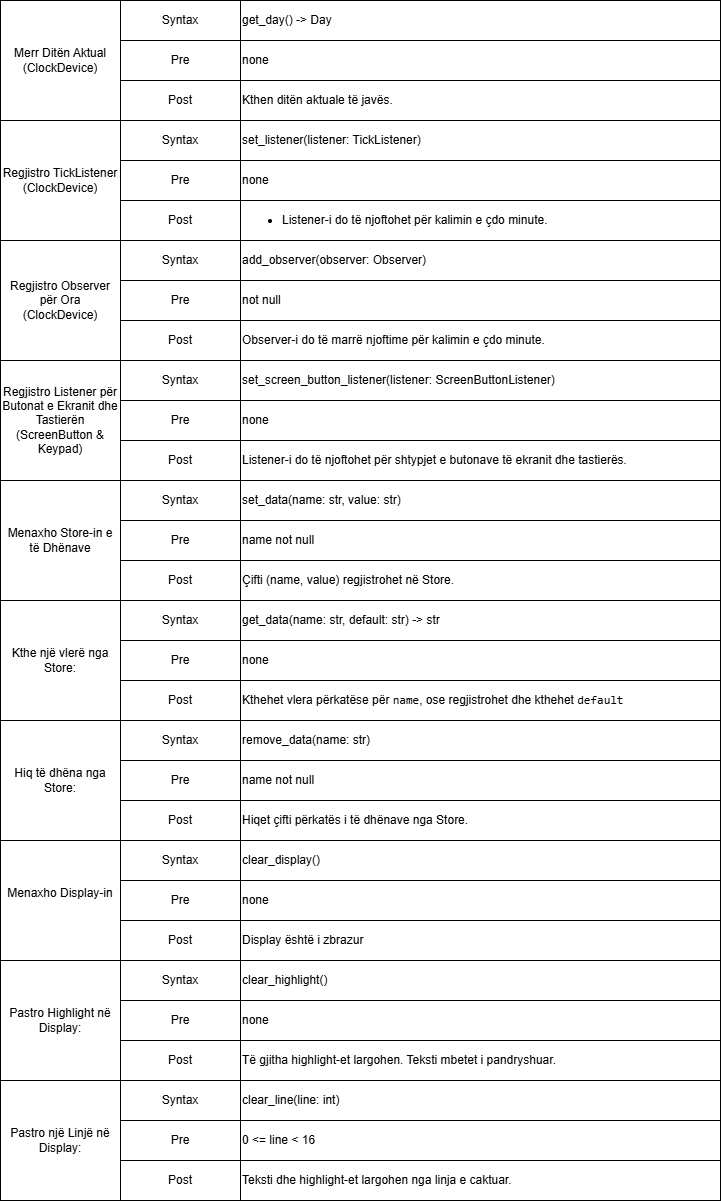
Roli i **SimTime** është gjithashtu subjekt në modelin **Observer**. Nëse do të ishte një invoker në modelin **Command**, atëherë mund të kishte vetëm një reaktor të vetëm. Megjithatë, disa komponentë në simulim, si dhe **ClockDevice** në shtresën **Device Interface**, duhet të njoftohen për kalimin e kohës së simuluar.

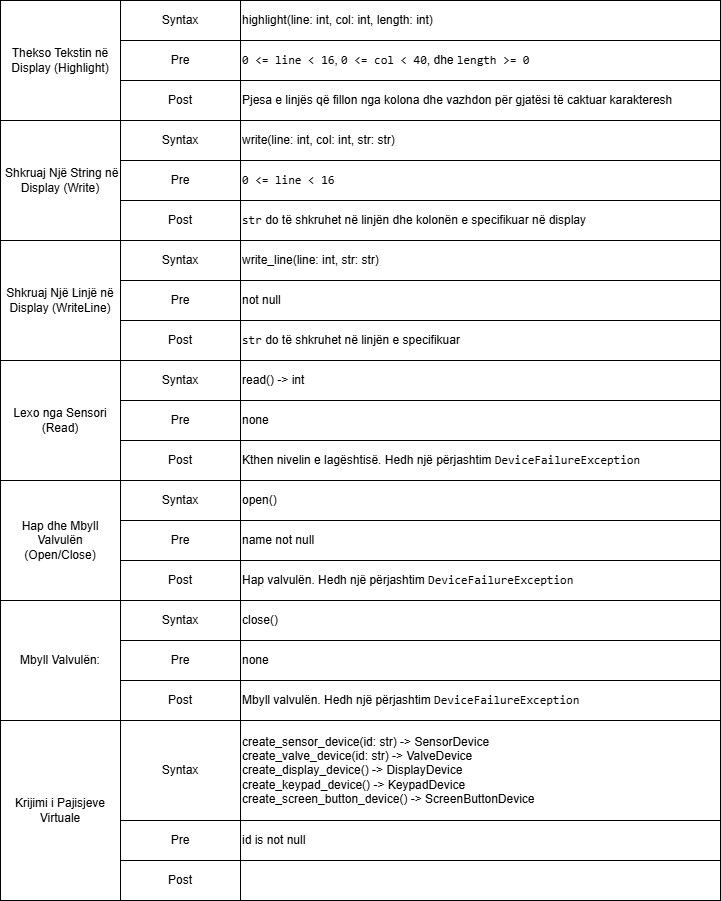
2.4 Device Interface Layer Decomposition

Shtresa **Device Interface** ofron pajisje virtuale për të fshehur entitetet reale ose të simuluara të jashtme që manipulohen nga programi. Shkrimi i programit për të ndërvepruar vetëm me pajisje virtuale e bën sistemin e Menaxhimit të Punonjësve shumë të konfigurueshëm. Pajisjet virtuale të ofruara në këtë shtresë janë të ngjashme me ato të paraqitura në figurën e ofruar:









2.4.3 Device Interface Layer Design Rationale

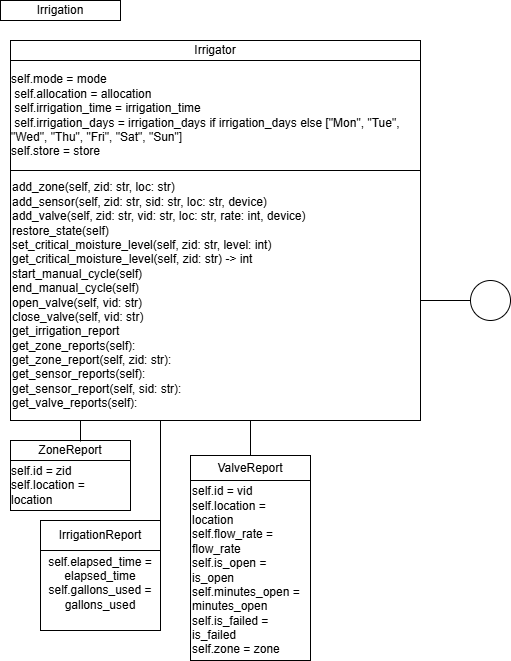
Versionet e tjera të kësaj shtrese nuk përfshinin **DeviceFactory**. **DeviceFactory** është pjesë e modelit **Abstract Factory**, i cili fsheh detajet e konfigurimit të pajisjeve virtuale brenda fabrikave konkrete dhe e bën të lehtë ndryshimin e konfigurimit. Alternativat që përfshijnë modelin **Abstract Factory** janë dukshëm më të mira.

Një pyetje e diskutuar disa herë ishte si të menaxhohej ora. A duhet të ketë një entitet të veçantë për **Clock** dhe **ClockDevice**? Ora ofron shërbime të kohës dhe njoftimit për klientët, ndërsa **ClockDevice** shërben si ndërfaqe për pajisjet reale ose të simuluara. Përfundimisht, u zgjodh alternativa me dy entitete për të ruajtur uniformitetin me modulet e tjera të shtresës **Device Interface**.

Sa i përket vendosjes së **Clock**, megjithëse ofron shërbime të kohës dhe njoftimit, u vendos që të jetë pjesë e shtresës **Device Interface**, pasi konsiderohet si një lloj pajisjeje dhe nuk përshtatet mirë në shtresat e tjera.

2.5 Irrigation Layer Decomposition

Shtresa **Irrigation** është thelbi i funksionalitetit të programit, duke menaxhuar sistemin kryesor të ujitjes. Struktura e saj arkitekturore është e thjeshtë: përfshin një modul **façade** që ofron operacione për konfigurim, vendosjen e parametrave të ujitjes dhe kontrollin manual të saj. Operacionet varen nga një lloj enumerimi dhe disa klasa që përmbajnë të dhëna për gjendjen e programit.



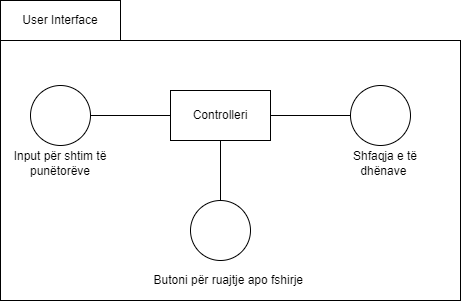
2.5.3 Irrigation Layer Design Rationale

Shumë alternativa të projektimit ekspozojnë strukturën e brendshme të shtresës Irrigation duke bërë që klasat Zone, Sensor, dhe Valve të jenë pjesë e ndërfaqes së shtresës. Kjo e thjeshton klasën Irrigator dhe eliminon nevojën për klasat e shumta të raporteve që kthejnë të dhëna për konfigurimin, gjendjen e ujitjes dhe pajisjet.

Megjithatë, këto alternativa kanë dy probleme: ekspozimi i strukturës së brendshme e bën më të vështirë ndryshimin në të ardhmen dhe kufizon aftësinë për të zbatuar kufizime mbi sjelljen e ujitjes. Për shembull, alternativat e mëparshme nuk lejojnë hapjen ose mbylljen manuale të valvulave përveçse në mënyrën automatike. Aksesimi i drejtpërdrejtë i valvulave do ta bënte të vështirë zbatimin e këtij kufizimi. Zgjedhja më e mirë duket alternativa që promovon fshehjen e informacionit dhe kontrollin mbi thjeshtësinë.

2.6 User Interface Layer Decomposition

Shtresa User Interface menaxhon ndërveprimin e përdoruesit përmes një ekrani tekstual monokrom dhe inputeve nga një tastierë me 12 taste dhe 8 butona ekrani. Kjo shtresë ka një UIController që ekzekuton një makinë të gjendjes të bazuar në hartën e dialogut të ndërfaqes. Kontrolluesi shpërndan inputin nga tastiera, butonat e ekranit dhe ora te ekrani aktiv aktualisht.

****

2.6.1

User Interface Layer Module Responsibilities

|  |  |
| --- | --- |
| Module | Responsibilities |
| Kontrolleri | Kontrolleri e bene ekzekutimin e te dhenave qe hyne, dhe shfrytezon butonin per ruajtje dhe fshirjen per ekzekutimin e tyre ne fund bene shfaqjen |

Table: User Interface Layer Module Responsibilities

2.6.2 User Interface Layer Interface Specifications

|  |  |  |
| --- | --- | --- |
| Krijimi i User Interface-it | Syntax: |  |
| Pre: |  |
| Post: |  |

Table: User Interface Layer Interface Specifications

2.6.3 User Interface Layer Design Rationale

Pavarësisht se qëllimi është që kjo shtresë të implementojë një makinë gjendjeje që realizon hartën e dialogut të ndërfaqes, përshkrimi i arkitekturës është aq i përgjithshëm saqë çdo lloj implementimi është i mundur: UIController mund të trajtohet si një façadë që fsheh çdo lloj implementimi alternativ. Kjo është ndoshta arsyeja më e mirë për të preferuar këtë alternativë arkitekturore. Alternativa të tjera do të siguronin më shumë detaje që do të kufizonin mundësitë për projektimin e detajuar.

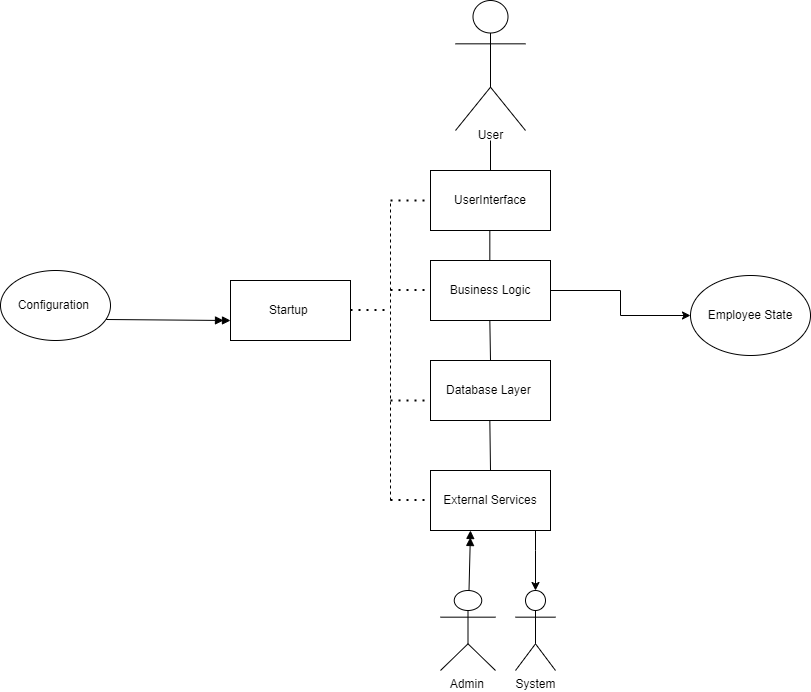
1 Mid-Level Design Models

Modelet e dizajnit të nivelit të mesëm zhvillojnë secilën shtresë arkitekturore dhe janë të organizuara në këtë seksion sipas shtresave.

1.1 Struktura Statike e Shtresës Startup

Shtresa Startup është përgjegjëse për inicializimin e programit si pjesë e një produkti të implementuar ose si një simulim në web, duke e konfiguruar programin sipas harduerit që është në dispozicion, dhe duke rikthyer gjendjen e mëparshme të programit. Shtresa Startup ka strukturën statike të treguar në figurën B-11-1.

**2.7 Run time Components**



Ky diagram përshkruan një arkitekturë të strukturuar dhe të definuar për sistemin e menaxhimit të punëtorëve, dhe çdo modul ka një funksion specific për operacionet e sistemit.

Configuration bënë inicimin e sistemit tek shtresa Startup.

Pastaj përdoruesi ndervepron më User Interface dhe kërkesat e tij shkojnë tek Business Logic.

Business Logic i merr kerkesat dhe bashkepunon me shtresen e Databazes për ruajten dhe kthimin e të dhënave.

Pastaj nëse kemi nevoj për sherbime të jashtme sic janë pagesat përdoret External Services.

Dhe e gjithë kjo gjendje aktuale ruhet ne Employee State.

3. Mapping Between Models

Kjo tabelë përfshine

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **User Interface** | **Business Logic** | **Database Layer** | **External Services** |
| **Startup** |  |  |  |  |
| **User Interface** |  |  |  |  |
| **Business Logic** |  |  |  |  |
| **Database Layer** |  |  |  |  |

|  |  |
| --- | --- |
| **Runtime Component** | **Modules** |
| Startup | In Startup starts the configuration of modules. |
| User Interface | Displays forms dashboards for interaction. |
| Business Logic | Proceesses employee data, calculates salaries, and manage attendance. |
| Database Layer | Handles database operations (CRUD) for all modules. |
| External Services | Sends notifications for payroll services etc. |

**4. Architectural Design Rationale**

Shtresat në këtë program janë të dizajnuara për të siguruar konfigurueshmëri, një nga atributet kryesore të sistemit. Softueri thelbësor, kryesisht përmbajtja e shtresës për menaxhimin e punëtorëve, duhet të funksionojë pa ndryshime në simulim ose produkt të implementuar, kështu që programi mund të modifikojë ndërveprimet me mjedisin e tij. Kjo përfshin elemente si koha e simuluar dhe ruajtja e të dhënave të punëtorëve, të cilat mund të ndryshojnë në varësi të mjedisit.

Për të menaxhuar ndarjen e pajisjeve dhe resurseve të ndryshme të mjedisit, është e nevojshme të ketë një shtresë ndërfaqe pajisjesh (Device Interface Layer), e cila mbulon mjedisin e simulimit dhe pajisjet në një sistem të implementuar. Ky ndarje siguron që programi të mbetet fleksibël dhe i thjeshtë për t’u modifikuar në të ardhmen.

Për ndërfaqen e përdoruesit, është më efikase që ajo të realizohet në një komponent të pavarur që përdor pajisje virtuale, pasi ky qasje është më e qëndrueshme dhe mund të menaxhojë ndryshimet mes mjediseve të ndryshme.

Në fund, për të menaxhuar kompleksitetin e konfigurimeve, shtresa e startup-it është përgjegjëse për inicializimin dhe konfigurimin e programit, duke e bërë më të lehtë për modifikim dhe testim.

