Management in camine

Echipa:

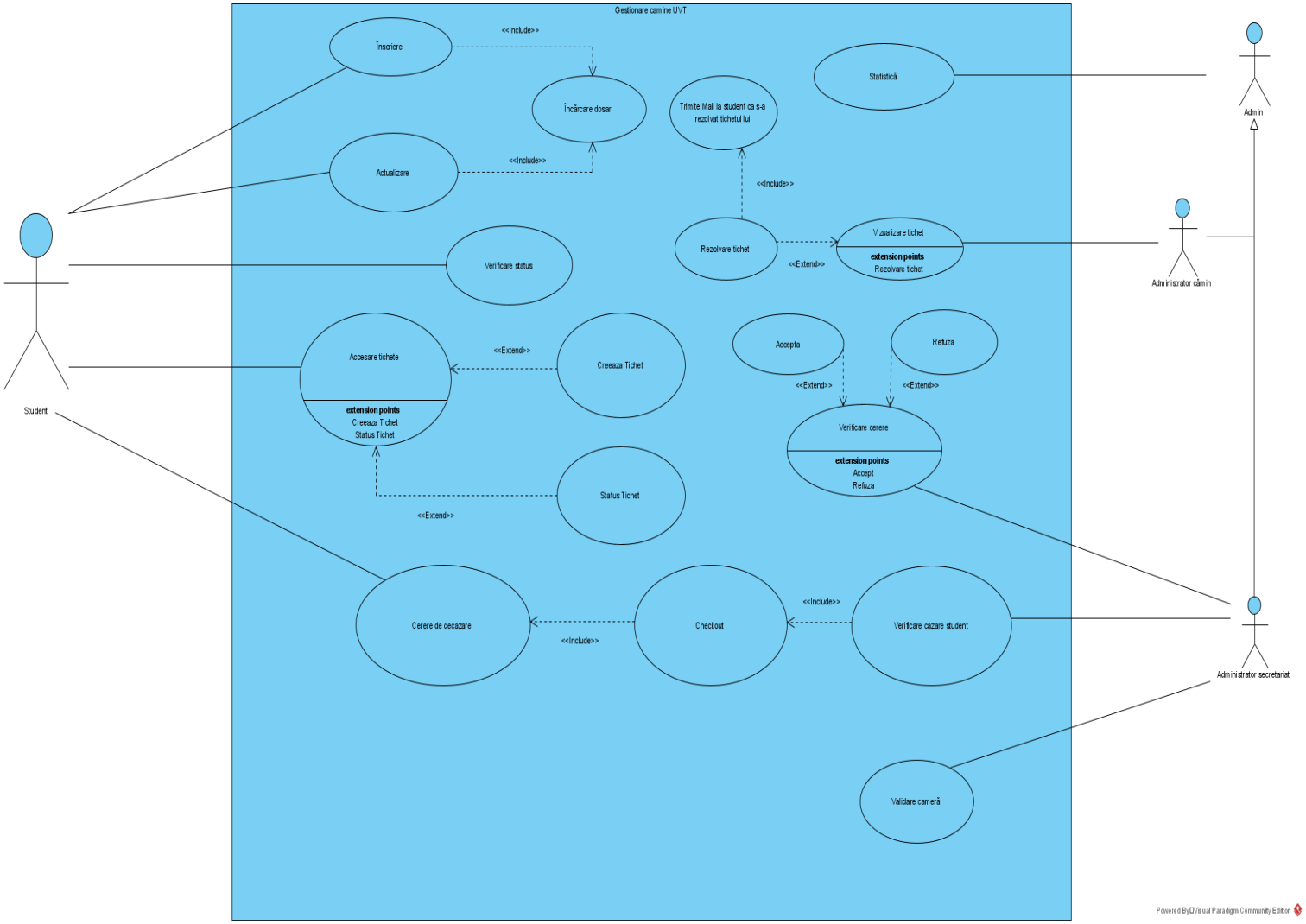
- Miklo Benjámin
- Németh Tímea Sarah
- Pintea Lorena
- Florin Mărut

Contents

Descriere:

Crearea unei platforme legată de procesul de aplicare pentru un loc în cămine cât și pentru experiența pe care studenții o au în momentul în care sunt cazați.

Diagrama Cazurilor de Utilizare



Cerintele Sistemului**:**

1) Log in (Student / Admin)
2) Inscriere (Student)
3) Incarcare dosar (Student)
4) Actualizare dosar (Student)
5) Verificare status (Student)
6) Creeare tichete (Student)
7) Status tichete (Student)
8) Decazare (Student)
9) Statistica (Admin)
10) Vizualizare tichet (Student / Admin)
11) Rezolvare tichet (Admin)
12) Verificare cerere (Admin)
13) Acceptare / Refuzare candidat (Admin)
14) Verificare cazare student (Admin)
15) Validare camera (Admin)

MoSCoW - Prioritizare **(Must Have, Should Have, Could have, Won't have)**

Must Have

1. Log in (Student / Admin)
2. Inscriere (Student)
3. Incarcare dosar (Student)
4. Verificare cerere cazare (Admin)
5. Creeare tichet (Student)
6. Vizualizare + rezolvare tichet (Admin)
7. Verificare cerere (Admin)
8. Acceptare / Refuzare candidat (Admin)

Should Have

9. Decazare (Student)
10. Verificare status (Student)
11. Status tichet (Student)

Could Have

12. Verificare cazare student (Admin)
13. Statistica (Admin)

Won’t Have

14. Validare camera (Admin)
15. Actualizare dosar (Student)

Descrierea a doua cazuri de utilizare

Un caz de utilizare fără relații include/extend

-**Nume**: Statistică
-**Scurtă descriere**: Afișează datele agregate despre situația tichetelor, studenților, etc.
-**Condiție prealabilă**: Adminul trebuie să fie autorizat.
-**Postcondiție**: Datele sunt agregate.
-**Situații de eroare**: Utilizatorul nu este admin autorizat, datele nu sunt disponibile.
-**Starea sistemului în caz de eroare**: Datele nu sunt agregate.
-**Actori**: Admin
-**Declanșator**: Are nevoie de date agregate.
-**Proces standard**:

1) Adminul se conectează la sistem
2) Selectează datele pe care le vrea agregate
3) Sistemul confirmă că datele există
4) Adminul primește datele cerute

-**Procese alternative**:

1) Adminul se conectează la sistem
2) Selectează datele pe care le vrea agregate
3) Sisteml nu gaseste datele cerute
4) Adminul primeste un mesaj de eroare

Un caz de utilizare cu relații include și extend

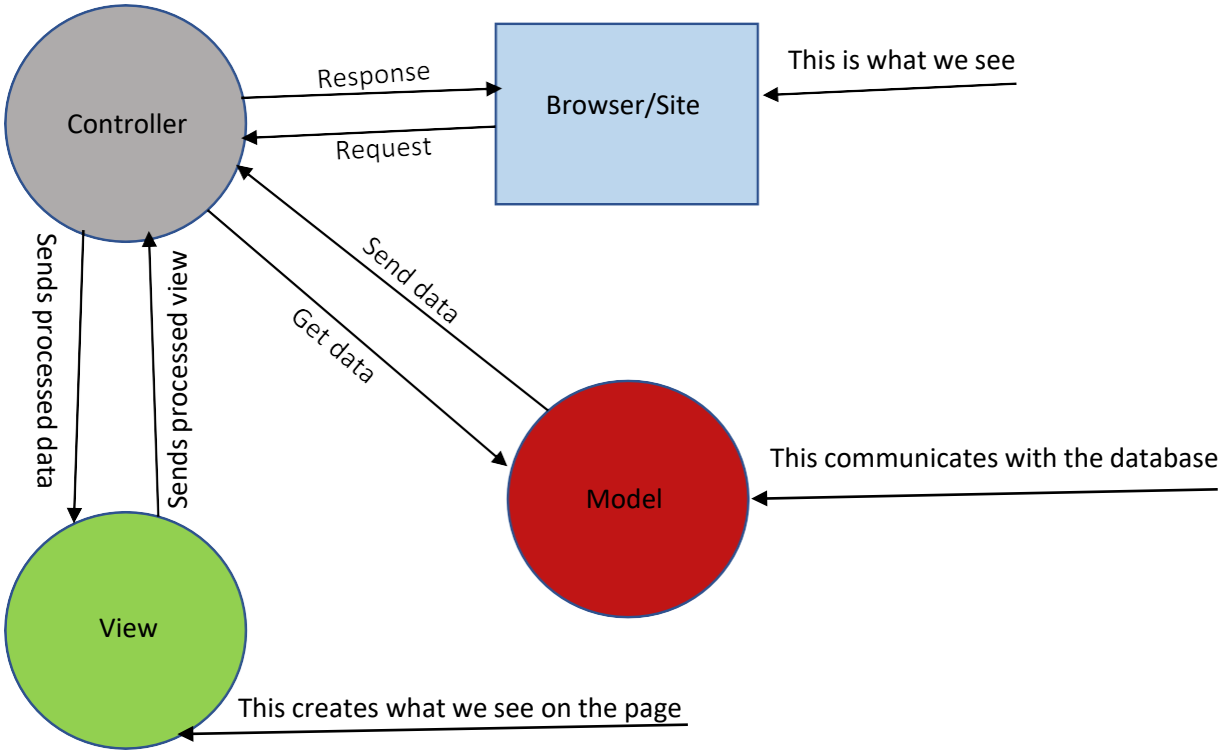
-**Nume**: Rezolvare tichet.
-**Scurtă** descriere: Adminul rezolva tichetul creat de student.
-**Condiție prealabilă**: Adminul trebuie să fie autorizat si sa fie admin de camin.
-**Postcondiție**: Tichetul studentului este rezolvat si primeste un mail cu aceasta informatie.
-**Situații de eroare**: Mail-ul studentului este gresit.
-**Starea sistemului în caz de eroare**: Studentul nu este notificat prin e-mail.
-**Actori**: Admin - Student
-**Declanșator**: Studentul creeaza un tichet si adminul vizualizeaza apoi rezolva tichetul.
-**Proces standard**:

1) Studentul creeaza un tichet
2) Adminul vizualizeaza tichetul
3) Adminul rezolva problema descrisa in tichet
4) Studentul este notificat prin e-mail ca problema lui a fost rezolvata.

-**Procese alternative**:

1) Studentul creeaza un tichet
2) Adminul vizualizeaza tichetul
3) Adminul rezolva problema descrisa in tichet
4) Studentul nu este notificat fiindca are e-mail gresit

Arhitectura Proiectului



De ce aceasta arhitectura?

E recomandat de a se folosi in proiecte realizate cu ASP .NET si e usor de lucrat cu el.

Descrierea Stilului MVC

Problema

Apare nevoia de modularizare a aplicatiei, de a delimita in mod clar partile componente pentru a putea fi usor modificate.

Context

In general, scopul multor computere este acela de a prelua informatii dintr-o anumita locatie, de a le prelucra dupa preferintele utilizatorului si, in cele din urma, de a le afisa utilizatorului.

Cea mai usoara metoda de a realiza o aplicatie care realizeaza aceste operatii este de a pune laolalta operatiile si de a le trata ca pe un tot.

Solutia

O solutie la aceasta problema este arhitectura Model-View-Controller(MVC) care separa partea de stocare a datelor de cea de prezentare si de prelucrare. Avem asadar trei clase distincte: Model (se ocupa de datele aplicatiei), View (transpune modelul intr-o interfata vizuala) si Controller (primeste input(butoane, formulare, etc.) de la utilizator si initiaza un raspuns).

Variante

HMVC (Hierarchical Model-View-Controller)
MVP (Model View Presenter)
MVVM (Model View ViewModel)

Exemple

MVC este des intalnit in in aplicatii web, aplicatii mobile

Diagrama de clase