Sistem software pentru prelucrarea si gestionarea comenzilor intr-un restaurant

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc164019944)

[1.1 Scopul documentului 1](#_Toc164019945)

[2. Prezentare generală și abordări de proiectare 2](#_Toc164019946)

[2.1 Prezentare generală 2](#_Toc164019947)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc164019948)

[2.2.1 Presupuneri 2](#_Toc164019949)

[2.2.2 Constrângeri 2](#_Toc164019950)

[2.2.3 Riscuri 2](#_Toc164019951)

[3. Considerațiii de proiectare 4](#_Toc164019952)

[3.1 Obiective și linii directoare (ghiduri). 4](#_Toc164019953)

[3.2 Metode de dezvoltare 5](#_Toc164019954)

[3.3 Strategii de arhitectură 5](#_Toc164019955)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc164019956)

[4.1 Vedere logică 7](#_Toc164019957)

[4.2 Arhitectură hardware 8](#_Toc164019958)

[4.3 Arhitectură software 8](#_Toc164019959)

[4.4 Arhitectura informațiilor 9](#_Toc164019960)

[4.5 Arhitectura de comunicații interne 9](#_Toc164019961)

[4.6 Diagrama de arhitectură a sistemului 10](#_Toc164019962)

[5. Proiectarea sistemului 11](#_Toc164019963)

[5.1 Proiectarea bazei de date 11](#_Toc164019964)

[5.2 Interfețe utilizator 11](#_Toc164019965)

[5.3 Proiectarea interfețelor cu utilizatorul 12](#_Toc164019966)

[6. Scenarii de utilizare 14](#_Toc164019967)

[7. Proiectare de detaliu 15](#_Toc164019968)

[7.1 Proiectare software de deatliu 15](#_Toc164019969)

## Introducere

Numele complet al proiectului: Sistem Software pentru Preluarea și Gestionarea Comenzilor în Restaurant

Acronimul proiectului: SSGCR

Situația actuală: În prezent, majoritatea restaurantelor utilizează procese manuale sau soluții software generice pentru gestionarea comenzilor, ceea ce poate duce la erori și întârzieri în servirea clienților.

Situația propusă: Implementarea SSGCR va aduce în restaurant un sistem software dedicat, specializat în preluarea și gestionarea eficientă a comenzilor. Acest sistem va facilita fluxul de lucru, va reduce timpul de procesare a comenzilor și va îmbunătăți experiența clienților.

Evoluția așteptată a documentului: Documentul de Proiectare a Sistemului va evolua pe măsură ce se progresează în fazele de proiectare și dezvoltare a sistemului. Inițial, va furniza o prezentare generală a arhitecturii sistemului și a obiectivelor de proiectare, urmând să detalieze specificațiile tehnice și funcționale ale fiecărei componente a sistemului.

Considerații referitoare la securitate și confidențialitate: Documentul de Proiectare a Sistemului va include măsuri adecvate pentru protejarea datelor sensibile ale clienților și ale restaurantului, respectând toate reglementările și standardele legale privind confidențialitatea și protecția datelor. De asemenea, va identifica și propune soluții pentru gestionarea riscurilor potențiale de securitate.

### Scopul documentului

Scopul Documentului de Proiectare a Sistemului (SDD) în cadrul proiectului "Sistem Software pentru Preluarea și Gestionarea Comenzilor în Restaurant" (SSGCR) este de a furniza o structură bine definită și detaliată a arhitecturii și designului sistemului propus. Acest document este esențial pentru a ghida echipa de dezvoltare în implementarea eficientă a sistemului și pentru a asigura o înțelegere comună a modului în care componentele sistemului vor funcționa împreună.

SDD-ul va fi dezvoltat incremental și iterativ pe parcursul ciclului de viață al dezvoltării sistemului SSGCR, pentru a reflecta nevoile specifice ale proiectului și metodologia de dezvoltare utilizată. Principalele obiective ale acestui document sunt de a oferi un cadru clar pentru implementarea sistemului, de a asigura coerența și coeziunea între diferitele componente și funcționalități, și de a servi drept referință pentru toți membrii echipei de proiect, inclusiv managerul de proiect, echipa de proiect și echipa de dezvoltare.

În plus, SDD-ul va conține detalii relevante despre arhitectura sistemului, interfețele, modelele de date, componente software, integrarea sistemului cu alte sisteme sau servicii, și orice alte aspecte tehnice și de proiectare necesare pentru implementarea și operaționalizarea sistemului SSGCR. Acest document va fi o resursă esențială pentru asigurarea alinierii între cerințele sistemului, arhitectura propusă și implementarea finală, contribuind astfel la succesul și eficiența proiectului.

## Prezentare generală și abordări de proiectare

### Prezentare generală

Contextul sistemului este reprezentat de nevoia unui restaurant de a implementa un sistem software eficient pentru preluarea și gestionarea comenzilor. Scopul acestui proiect este să ofere o soluție care să optimizeze procesele interne ale restaurantului și să îmbunătățească experiența clienților.

Arhitectura software-ului: Software-ul va fi structurat pe mai multe niveluri, inclusiv nivelul de interfață cu utilizatorul, nivelul de logică a aplicației și nivelul de acces la date. Această separare a responsabilităților va permite o dezvoltare mai modulară și o gestionare mai ușoară a codului.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

Presupunerile și dependențele legate de sistemul "Sistem Software pentru Preluarea și Gestionarea Comenzilor în Restaurant" includ următoarele:

1. Dependențe de hardware
2. Dependențe de software
3. Dependențe de sisteme de operare
4. Caracteristicile utilizatorilor finali
5. Modificări viitoare ale funcționalității

#### Constrângeri

În cadrul proiectului "Sistem Software pentru Preluarea și Gestionarea Comenzilor în Restaurant", există mai multe limitări și constrângeri care pot influența designul hardware-ului, software-ului și comunicațiilor sistemului:

* Disponibilitatea și stabilitatea conexiunii la internet
* Capacitatea hardware-ului
* Cerințele de securitate și confidențialitate
* Cerințele de performanță
* Interoperabilitatea cu alte sisteme și servicii

Aceste limitări și constrângeri trebuie luate în considerare în procesul de proiectare și implementare a sistemului, pentru a asigura că acesta își îndeplinește obiectivele în mod eficient și conform cu cerințele și așteptările stabilite.

#### Riscuri

1. Riscul de pierdere a datelor: Existența unei conexiuni la internet instabile sau a echipamentului hardware defect poate duce la pierderea datelor comenzilor sau a altor informații importante. Pentru a reduce acest risc, se va implementa o strategie de backup a datelor în cloud, astfel încât să se asigure că informațiile critice sunt protejate și disponibile în caz de necesitate.
2. Riscul de securitate cibernetică: Sistemul poate fi expus la riscul de atacuri cibernetice, cum ar fi hacking-ul sau furtul de date. Pentru a contracara aceste amenințări, se va implementa o serie de măsuri de securitate, cum ar fi autentificarea cu doi factori, criptarea datelor sensibile și monitorizarea activității suspecte în sistem.
3. Riscul de performanță insuficientă: Un volum mare de comenzi sau o arhitectură inadecvată a sistemului pot duce la performanță insuficientă sau la întârzieri în preluarea și procesarea comenzilor. Pentru a reduce acest risc, se vor efectua teste de sarcină pentru a evalua capacitățile sistemului și se vor implementa optimizări pentru îmbunătățirea performanței.
4. Riscul de neconformitate cu reglementările legale: Sistemul trebuie să respecte reglementările legale privind protecția datelor și confidențialitatea informațiilor personale ale clienților. Pentru a reduce acest risc, se va efectua o analiză detaliată a cerințelor legale relevante și se vor implementa măsuri adecvate pentru conformitate.
5. Riscul de neinteroperabilitate: Incompatibilitatea cu alte sisteme sau servicii utilizate de restaurant poate afecta capacitatea sistemului de a funcționa în mod corespunzător. Pentru a reduce acest risc, se vor efectua teste de integrare și se va colabora cu furnizorii altor sisteme pentru a asigura interoperabilitatea adecvată.

## Considerațiii de proiectare

1. Analiza cerințelor și nevoilor utilizatorilor: Este crucial să înțelegem în detaliu cerințele și nevoile clienților, precum și ale personalului restaurantului, pentru a dezvolta o soluție care să satisfacă cu succes aceste cerințe. Această analiză ar trebui să includă o varietate de factori, cum ar fi preferințele utilizatorilor, fluxurile de lucru curente din restaurant și obstacolele cu care se confruntă.
2. Evaluarea mediului tehnologic și a infrastructurii existente: Trebuie să evaluăm infrastructura hardware și software existentă din restaurant și să determinăm dacă există necesitatea de a actualiza sau de a înlocui anumite componente pentru a asigura compatibilitatea și eficiența sistemului nou propus.
3. Stabilirea unui buget și a resurselor disponibile: Este important să stabilim un buget adecvat și să evaluăm resursele umane și financiare disponibile pentru dezvoltarea și implementarea sistemului. Acest lucru ne va ajuta să gestionăm așteptările și să prioritizăm funcționalitățile în funcție de resursele disponibile.
4. Analiza riscurilor și a reglementărilor legale: Trebuie să identificăm și să evaluăm riscurile potențiale asociate cu implementarea sistemului, precum și să ne asigurăm că respectăm reglementările legale relevante, cum ar fi cele referitoare la protecția datelor și la securitatea informațiilor personale ale clienților.
5. Stabilirea obiectivelor de proiectare și a criteriilor de succes: Trebuie să stabilim obiective clare de proiectare și să dezvoltăm criterii de succes pentru evaluarea soluției noastre de design. Aceste obiective ar trebui să fie cuantificabile și măsurabile, permițându-ne să evaluăm eficacitatea și eficiența sistemului.

### Obiective și linii directoare (ghiduri).

Pentru "Sistemul Software pentru Preluarea și Gestionarea Comenzilor în Restaurant", următoarele obiective, linii directoare și principii domină sau sunt încorporate în designul sistemului și al software-ului:

1. Performanță și scalabilitate: Unul dintre principalele obiective este asigurarea unei performanțe ridicate și a scalabilității sistemului pentru a putea gestiona un volum mare de comenzi și utilizatori simultan, fără a compromite viteza sau eficiența.
2. Ușurința în utilizare: Interfața cu utilizatorul trebuie să fie intuitivă și ușor de folosit, astfel încât clienții și personalul restaurantului să poată naviga și să folosească sistemul fără dificultăți sau instruiri complexe.
3. Securitate și confidențialitate: Se acordă o atenție deosebită securității și confidențialității datelor, cu implementarea măsurilor adecvate de autentificare, criptare și control al accesului pentru a proteja informațiile sensibile ale clienților și ale afacerii.
4. Flexibilitate și extensibilitate: Sistemul trebuie să fie flexibil și extensibil, permițând adăugarea și modificarea ulterioară a funcționalităților în funcție de evoluția nevoilor și cerințelor afacerii și ale utilizatorilor.
5. Consistență și standardizare: Se aplică linii directoare și convenții de codare pentru a asigura consistența și standardizarea în întregul sistem, ceea ce facilitează înțelegerea și mentenanța acestuia de către membrii echipei de dezvoltare.
6. Interoperabilitate și integrare: Se prioritizează interoperabilitatea și integrarea cu alte sisteme și servicii utilizate în cadrul restaurantului, pentru a asigura o experiență fluidă și o comunicare eficientă între diferitele componente software.
7. Eficiență și optimizare: Se urmărește optimizarea performanței și utilizarea eficientă a resurselor hardware și software disponibile, pentru a asigura o funcționare cât mai fluidă și fără probleme a sistemului.

### Metode de dezvoltare

Pentru designul sistemului și al software-ului "Sistemului Software pentru Preluarea și Gestionarea Comenzilor în Restaurant", se va adopta o abordare orientată pe obiecte, combinată cu tehnici moderne de dezvoltare web. Acest lucru implică utilizarea unor structuri de date și a unor interfețe bine definite pentru a organiza și a gestiona funcționalitățile sistemului. De asemenea, se va folosi programarea reactivă pentru a asigura o interfață utilizator fluidă și o comunicare eficientă între componentele software.

UML (Unified Modeling Language) va fi utilizat pentru a documenta și a comunica designul sistemului, oferind o modalitate standardizată și clară de a reprezenta arhitectura și interacțiunile între componente.

În ceea ce privește contingentele, este posibil să apară schimbări în arhitectura sau cerințele sistemului pe parcursul dezvoltării, mai ales în contextul unui mediu în continuă schimbare, precum cel al restaurantelor. Pentru a face față acestor situații, vom adopta o abordare agilă, permițând flexibilitate și adaptabilitate în cadrul procesului de dezvoltare. De asemenea, vom documenta și vom comunica transparent aceste schimbări către toți membrii echipei și către părțile interesate pentru a asigura o înțelegere comună și pentru a ajusta planurile și prioritățile în consecință.

### Strategii de arhitectură

Pentru "Sistemul Software pentru Preluarea și Gestionarea Comenzilor în Restaurant", următoarele decizii și strategii de design afectează organizarea generală a sistemului și structurile sale de nivel superior:

1. Utilizarea framework-ului React pentru frontend: Am decis să folosim framework-ul React datorită eficienței sale în dezvoltarea interfețelor utilizator moderne și bogate în funcționalități. React ne permite să organizăm componentele UI în mod modular și să le reutilizăm eficient, contribuind la o dezvoltare rapidă și ușoară a interfeței utilizator.
2. Implementarea serviciilor cloud pentru gestionarea datelor: Pentru a asigura scalabilitatea și disponibilitatea sistemului, am decis să folosim servicii cloud precum Firebase pentru gestionarea datelor. Aceasta ne permite să beneficiem de infrastructura scalabilă și de baze de date gestionate, eliminând necesitatea de a dezvolta și de a menține o infrastructură proprie.
3. Adoptarea unei abordări reactive pentru comunicarea între client și server: Pentru a asigura o comunicare eficientă și reactivă între clientul restaurantului și server, vom folosi tehnologii precum WebSocket-uri și RxJS pentru a gestiona schimbul de date în timp real între diferitele componente ale sistemului.
4. Utilizarea unei arhitecturi bazate pe microservicii: Pentru a facilita scalabilitatea și flexibilitatea sistemului, vom adopta o arhitectură bazată pe microservicii, în care diferitele funcționalități ale sistemului sunt împărțite în servicii independente, cu interfețe bine definite. Acest lucru ne permite să dezvoltăm, să testăm și să implementăm fiecare serviciu în mod independent, facilitând extinderea și întreținerea ulterioară a sistemului.

## Arhitectura Sistemului și Proiectarea Arhitecturii

Arhitectura sistemului software pentru gestionarea și prelucrarea comenzilor într-un restaurant, cu aplicație web pentru digitalizarea procesului de plasare a comenzilor, utilizează tehnologii moderne precum HTML, CSS, JavaScript, React și Firebase. Interfața cu clientul, dezvoltată în React, oferă o experiență interactivă utilizatorilor, inclusiv funcționalități precum vizualizarea meniului, adăugarea produselor în coș și finalizarea comenzilor. Această interfață include acum și o pagină de autentificare, gestionată de componente React, care solicită utilizatorilor să introducă credențialele (nume de utilizator și parolă).

Serverul aplicației, construit în React și Firebase, integrează serviciul Firebase Authentication pentru gestionarea procesului de autentificare a utilizatorilor. Funcționalitățile avansate ale aplicației, precum plasarea comenzilor și vizualizarea istoricului, sunt accesibile doar utilizatorilor autentificați. Baza de date, Firebase Cloud Firestore, a fost extinsă pentru a personaliza accesul la date în funcție de utilizatorii autentificați, astfel încât fiecare client să aibă acces doar la propriul istoric de comenzi și informații personale.

În ceea ce privește sistemul de procesare a plăților, acesta integrează Firebase Authentication pentru a asigura securitatea și accesul doar pentru utilizatorii autentificați. Utilizează servicii precum Firebase Functions pentru gestionarea securizată și procesarea plăților, în concordanță cu autentificarea efectuată.

Sistemul de comunicare cu bucătăria, bazat pe Firebase și React, actualizează în timp real starea comenzilor în funcție de feedback-ul primit de la bucătărie. În plus, permite vizualizarea stării comenzilor anterioare doar pentru utilizatorii autentificați. Aceste adăugari de funcționalități de autentificare îmbunătățesc securitatea și personalizarea experienței utilizatorilor, asigurând că accesul la date și funcționalități este adaptat identității fiecărui utilizator.

Sistemul software pentru gestionarea și prelucrarea comenzilor într-un restaurant, cu aplicație web pentru digitalizarea procesului de plasare a comenzilor, a fost dezmembrat în mai multe componente sau subsisteme, fiecare cu roluri și responsabilități bine definite, contribuind astfel la funcționarea eficientă a întregului sistem.

Unul dintre principalele subsisteme este reprezentat de Interfața cu Clientul (Client-Side Component), responsabilă pentru interacțiunea directă cu utilizatorii. Aceasta include funcționalități precum afișarea meniului, adăugarea produselor în coș și finalizarea comenzilor. Colaborează strâns cu serverul aplicației pentru a transmite comenzile și pentru a primi actualizări privind starea meselor.

La nivel de server, avem Serverul Aplicației (Server-Side Component), care își asumă sarcini cruciale precum procesarea comenzilor, validarea datelor și gestionarea stării meselor și a meniului. Interacționează în mod activ cu baza de date pentru a accesa și actualiza informațiile necesare, precum și cu sistemul de procesare a plăților pentru finalizarea tranzacțiilor financiare.

Un alt element esențial al sistemului este Baza de Date (Data Storage Component), care asigură stocarea persistentă a informațiilor despre meniu, starea meselor și istoricul comenzilor. Aceasta este accesată și actualizată de către server pentru a asigura consistența datelor în întregul sistem.

Alegerea acestei decompuneri a sistemului a fost motivată de dorința de a separa eficient responsabilitățile și de a facilita dezvoltarea modulară și scalabilitatea. Această abordare permite gestionarea eficientă a interacțiunilor între componente, asigurând un grad optim de coeziune și cuplaj redus. Alte propuneri de decompunere au fost luate în considerare, dar această structură modulară a fost preferată pentru a susține extensibilitatea și adaptabilitatea sistemului la viitoarele evoluții.

În cadrul arhitecturii sistemului software pentru gestionarea comenzilor în restaurant, există oportunități de aplicare a tiparelor de proiectare pentru a îmbunătăți structura și eficiența codului. Un astfel de exemplu ar putea fi utilizarea tiparului de proiectare Observator (Observer Pattern) pentru gestionarea actualizărilor stării comenzilor în timp real.

Observator (Observer Pattern) pentru Actualizările în Timp Real:

Un tipar Observator poate fi aplicat în implementarea sistemului de comunicare cu bucătăria și a interfeței cu clientul. Astfel, bucătăria și interfața cu clientul pot fi văzute ca obiecte observabile, iar componente precum starea comenzilor sau statusul preparării pot fi observatori. Atunci când starea comenzii se schimbă (de exemplu, o comandă este pregătită în bucătărie), observatorii (interfața cu clientul) sunt notificați, permițând actualizarea în timp real a informațiilor afișate către utilizator.

Justificare: Această alegere se bazează pe nevoia de a actualiza starea comenzilor în timp real, fără a depinde de interogări periodice către server. Observatorul oferă o modalitate eficientă și dezcuplată de a notifica diverse părți ale sistemului despre schimbări de stare, asigurând coeziunea și lăsând loc pentru extensibilitate.

Fabrica (Factory Pattern) pentru Crearea Componentelor Interfeței cu Clientul:

Pentru crearea interfeței cu clientul, poate fi utilizat un tipar de proiectare Fabrica, care permite crearea de obiecte concrete (de exemplu, butoane sau elemente de meniu) fără a specifica clasa exactă a obiectului care urmează să fie creat.

Justificare: Utilizarea tiparului Fabrica aduce beneficii în modularitatea și extensibilitatea codului, deoarece adăugarea sau schimbarea tipurilor de componente ale interfeței cu clientul se realizează fără a modifica codul existent. Prin intermediul unei fabrici, pot fi create și gestionate diverse obiecte ale interfeței, facilitând astfel mentenanța și dezvoltarea viitoare a aplicației.

Aceste alegeri de tipare de proiectare au fost făcute pentru a îmbunătăți structura sistemului, facilitând întreținerea, extensibilitatea și o separare clară a responsabilităților în cadrul aplicației.

Această secțiune conturează designul arhitecturii sistemului și hardware-ului.

### Vedere logică

În cadrul proiectului "Sistemul Software pentru Preluarea și Gestionarea Comenzilor în Restaurant", logica relevantă este implementată în diferite componente și servicii software. Aceste vederi logice includ:

1. Componentele UI (Interfața Utilizatorului): Logica pentru gestionarea interacțiunii cu utilizatorii, cum ar fi procesarea evenimentelor de click, afișarea datelor și validarea intrărilor utilizatorului, este implementată în componentele UI, care sunt scrise folosind framework-ul React.
2. Serviciile Backend: Logica de afaceri, precum gestionarea comenzilor, autentificarea utilizatorilor, generarea de note de plată și gestionarea profilurilor utilizatorilor, este implementată în serviciile backend. Aceste servicii sunt scrise folosind Node.js și sunt găzduite pe o platformă cloud precum Firebase.
3. Baza de Date: Logica pentru gestionarea și persistența datelor este implementată în baza de date, care este gestionată de serviciile cloud precum Firestore de la Firebase. Aici sunt stocate informații despre utilizatori, meniuri, comenzi și alte date relevante pentru funcționarea sistemului.

### Arhitectură hardware

In cadrul proiectului "Sistemul Software pentru Preluarea și Gestionarea Comenzilor în Restaurant", sistemul de procesare este distribuit și include următoarele componente hardware:

1. Servere de prezentare: Acestea sunt serverele pe care rulează componentele frontend ale aplicației, care gestionează interacțiunea directă cu utilizatorii. Aceste servere pot fi distribuite în mai multe locații geografice pentru a asigura o experiență rapidă și scalabilă pentru utilizatori.
2. Servere de aplicație: Acestea sunt serverele pe care rulează serviciile backend ale aplicației, care gestionează logica de afaceri și comunicarea cu baza de date și alte servicii externe. Aceste servere sunt distribuite pentru a asigura redundanța și scalabilitatea sistemului.
3. Servere de bază de date: Acestea sunt serverele care găzduiesc baza de date a sistemului, care stochează informațiile despre utilizatori, meniuri, comenzi și alte date relevante. Aceste servere pot fi distribuite sau pot fi gestionate de servicii cloud precum Firebase.
4. Dispozitive periferice: În cadrul infrastructurii de rețea, putem include comutatoare, firewall-uri și load balancers pentru a asigura securitatea, performanța și redundanța sistemului. Firewall-urile și porturile de rețea sunt configurate pentru a permite accesul controlat la diferite componente ale sistemului și pentru a proteja împotriva amenințărilor externe.

Diagrama conectivității între aceste componente arată fluxurile de date între serverele frontend și backend, precum și interacțiunea cu baza de date și alte servicii externe. Resursele de capacitate ale procesorului, memoriei și stocării sunt estimate în funcție de cerințele de performanță și volumul de utilizare anticipat al sistemului.

### Arhitectură software

În cadrul proiectului "Sistemul Software pentru Preluarea și Gestionarea Comenzilor în Restaurant", următoarele componente software sunt implicate:

1. Frontend-ul Aplicației (Interfața Utilizatorului):

* Limbajul de programare: JavaScript
* Framework-ul: React
* Descriere: Componentele UI care permit utilizatorilor să interacționeze cu sistemul, inclusiv meniul, filtrarea, adăugarea de produse în coș și plasarea comenzilor.

1. Backend-ul Aplicației:

* Limbajul de programare: JavaScript (Node.js)
* Serviciul Cloud: Firebase
* Descriere: Serviciile backend care gestionează logica de afaceri, autentificarea utilizatorilor, gestionarea comenzilor și interacțiunea cu baza de date.

1. Baza de Date:

* Platforma: Firebase Firestore
* Descriere: Baza de date care stochează informațiile despre utilizatori, meniuri, comenzi și alte date relevante pentru funcționarea sistemului.

1. Instrumente și Utilitare:

* Visual Studio Code: IDE pentru dezvoltarea aplicației.
* Git și GitHub: Pentru gestionarea codului sursă și colaborarea în echipă.
* Firebase CLI: Pentru gestionarea și implementarea serviciilor Firebase.

1. Servicii de Comunicare:

* WebSocket-uri: Pentru comunicarea în timp real între frontend și backend.
* Notificări push Firebase: Pentru trimiterea de notificări către utilizatori în timp real.

### Arhitectura informațiilor

În cadrul sistemului de preluare și gestionare a comenzilor în restaurant, următoarele informații vor fi stocate:

1. Informații despre Utilizatori
2. Informații despre Meniu
3. Informații despre Comenzi
4. Informații despre Mese
5. Informații despre Autentificare și Sesiune

Aceste date vor fi colectate și stocate în baza de date a sistemului, Firebase Firestore. Informațiile despre utilizatori și comenzile lor pot fi considerate informații cu caracter sensibil, deoarece conțin date personale ale clienților, cum ar fi numele și adresa de email. Este important să se respecte standardele și reglementările privind protecția datelor pentru a asigura confidențialitatea și securitatea acestor informații.

### Arhitectura de comunicații interne

Pentru sistemul de preluare și gestionare a comenzilor în restaurant, se va implementa o arhitectură de rețea cu o topologie client-server. Iată o descriere detaliată a rețelei de comunicații:

1. Componente ale Rețelei:

* Clientul (Aplicația Client): Interfața prin care clienții restaurantului plasează comenzile. Va comunica direct cu serverul pentru a trimite și a primi informații despre comenzile plasate.
* Serverul (Aplicația Server): Găzduiește baza de date și logica de afaceri a sistemului. Primește comenzile de la clienți, le procesează și le stochează în baza de date. Trimite actualizări către clienți în timp real în legătură cu starea comenzilor.
* Baza de Date (Firebase Firestore): Stochează toate datele relevante pentru utilizatori, meniu, comenzi și mese.

1. Arhitectura de Comunicații:

* Comunicarea între client și server se va face prin intermediul unei conexiuni securizate, folosind HTTPS sau alte protocoale de comunicare criptate.
* Datele vor fi transmise în format JSON între client și server pentru a asigura interoperabilitatea și ușurința în procesarea acestora.
* Serverul va utiliza Firebase Firestore pentru a accesa și a actualiza baza de date. Comunicarea cu Firestore va fi gestionată prin intermediul API-ului Firebase oferit de Google.

1. Echipamentele Rețelei:

* Router: Pentru a gestiona traficul de date între client și server, precum și pentru a facilita conexiunea la internet.
* Server: O mașină dedicată sau un serviciu cloud care găzduiește aplicația serverului și baza de date.
* Dispozitive Client: Computere sau dispozitive mobile care rulează aplicația client.

### Diagrama de arhitectură a sistemului

A diagram of a software development

Description automatically generated

Figură 1 - diargama arhitectura software

## Proiectarea sistemului

### Proiectarea bazei de date

In cadrul sistemului de preluare și gestionare a comenzilor în restaurant, se utilizează o bază de date pentru a stoca și gestiona datele relevante:

Colectia „Preparate”:

* Descriere: contine toate categoriile de preparate gasite in meniu
* Campuri:
* Nume(String): Numele preparat
* Descriere(String): Informatii despre preparat
* Gramaj(Number): Cantitatea preparatului
* Pret(Number): Pretul preparatului

Colectia „Users”:

* Descriere: contine toti utilzatorii care si-au facut cont in aplicatie
* Campuri:
* ID(String): ID-ul utilizatorului
* Email(String): Emailul cu care utilizatorul si-a facut cont
* Parola(String): Parola contului
* PhotoURL(string): adresa imaginii de profil a contului utilizatorului

### Interfețe utilizator

Pentru proiectul "Sistemul Software pentru Prelucrarea și Gestionarea Comenzilor într-un Restaurant", putem identifica următoarele clase de utilizatori:

1. Clienți/Utilizatori Finali:

* Descriere: Aceștia sunt clienții care utilizează aplicația pentru a vizualiza meniul, a plasa comenzi și a efectua plăți.
* Estimare Număr Utilizatori: Numărul total de clienți care utilizează aplicația poate varia, dar se poate estima în funcție de popularitatea restaurantului și de traficul mediu.
* Număr Maxim Utilizatori Simultani: Acest număr poate fi determinat în funcție de capacitatea de gestionare a serverului și de infrastructura rețelei restaurantului.
* Interacțiuni: Vizualizarea meniului, adăugarea produselor în coș, plasarea comenzilor, efectuarea plăților.

1. Administratori ai Sistemului:

* Descriere: Aceștia sunt responsabili pentru administrarea și întreținerea sistemului software.
* Estimare Număr Utilizatori: De obicei, numărul administratorilor este mic și este restricționat la personalul IT al restaurantului sau la echipele de suport.
* Număr Maxim Utilizatori Simultani: Numărul maxim de administratori care accesează sistemul în același timp este limitat la nevoile de administrare și de gestionare a sistemului.
* Interacțiuni: Configurarea sistemului, gestionarea utilizatorilor și a permisiunilor, monitorizarea performanței sistemului.

Fiecare clasă de utilizatori are nevoi și interacțiuni specifice cu sistemul, iar estimările numărului de utilizatori și a numărului maxim de utilizatori simultani pot influența planificarea și proiectarea sistemului pentru a asigura o experiență optimă pentru toți utilizatorii implicați.

### Proiectarea interfețelor cu utilizatorul

A screenshot of a sign up form

Description automatically generated

Figură 2 - Pagina Signup

Pe aceasta pagina, utilizatorul isi poate crea cont, fie cu un email si o parola, fie cu un cont de Google deja existent.

A menu card with a green background

Description automatically generated

Figură 3 - pagina Home

Pe aceasta pagina a aplicatiei, utilizatorul poate vizualiza si interactiona cu preparatele din meniu, le poate selecta, cauta pentru a putea sa le comande ulterior.  
 Dupa ce comanda a fost plasata, clientul poate cere nota de plata, apasand pe butonul „Plateste”.

## Scenarii de utilizare

Sistemul software pentru prelucrarea și gestionarea comenzilor într-un restaurant are ca scop să ofere clienților o modalitate simplă și eficientă de a plasa și de a gestiona comenzile, precum și pentru personalul restaurantului de a monitoriza și a procesa comenzile într-un mod organizat și eficient. Sistemul integrează o interfață prietenoasă pentru utilizator pentru a vizualiza meniul, a plasa comenzi și a efectua plăți, împreună cu un portal de administrare pentru personalul restaurantului pentru gestionarea comenzilor, a meniului și a inventarului.

Scenarii operationale

Plasarea unei Comenzi de către un Client

1. Eveniment: Clientul deschide aplicația și navighează la secțiunea de meniu.
2. Acțiune: Clientul selectează articolele dorite din meniu și le adaugă în comanda sa.
3. Stimul: Clientul finalizează comanda și trimite solicitarea către sistem.
4. Informație: Sistemul primește comanda și o transmite către echipa restaurantului pentru procesare.
5. Interacțiune: Echipa restaurantului primește comanda pe softul dedicat și începe pregătirea articolelor.
6. Acțiune: După pregătire, chelnerii livrează comenzile la mesele corespunzătoare.

Cererea Notei de Plată de către Client

1. Eveniment: Clientul finalizează consumația și dorește să solicite nota de plată.
2. Acțiune: Clientul accesează aplicația și navighează la secțiunea de plată.
3. Stimul: Clientul selectează opțiunea "Cere Nota de Plată".
4. Informație: Sistemul primește solicitarea și o transmite către echipa restaurantului.
5. Interacțiune: Echipa restaurantului este notificată despre cererea de plată și pregătește nota.
6. Acțiune: Chelnerul aduce nota de plată la masa clientului.

Aceste scenarii descriu modul în care sistemul facilitează comanda și plata într-un restaurant, atât din perspectiva clientului, cât și din perspectiva personalului restaurantului.

## Proiectare de detaliu

1. Construirea și Integrarea Componentelor Hardware
2. Codificarea și Integrarea Componentelor Software
3. Interconectarea Segmentelor Hardware și Software

Pentru combinarea pachetelor separate externe într-un singur sistem:

* Evaluarea Pachetelor Externe: Examinarea și evaluarea pachetelor externe pentru a determina compatibilitatea și adecvarea lor pentru cerințele sistemului.
* Integrarea Pachetelor Externe: Integrarea pachetelor externe în arhitectura sistemului, asigurându-vă că sunt configurate și personalizate conform nevoilor specifice ale sistemului.
* Testarea Integrării Pachetelor: Testați funcționalitatea și interacțiunile pachetelor integrate cu restul sistemului pentru a verifica compatibilitatea și buna funcționare a acestora.
* Gestionarea Actualizărilor și a Patch-urilor: Monitorizarea și gestionarea actualizărilor și a patch-urilor pentru pachetele externe integrate pentru a menține securitatea și performanța sistemului.

### Proiectare software de deatliu

Identificator serviciu:

* Numele serviciului: Sistem software pentru gestionarea si prelucrarea comenzilor intr-un restaurant

Clasificare:

* Tipul de serviciu: Aplicație software

Definiție:

* Scopul specific: Serviciul SSGPCR este proiectat pentru a permite clienților să plaseze comenzi de la masa lor folosind o aplicație mobilă dedicată, să transmită aceste comenzi către bucătărie și să permită chelnerilor să gestioneze și să monitorizeze aceste comenzi pentru a asigura o experiență eficientă și plăcută pentru clienți.

Cerințe:

* Cerinte functionale:
* Permite utilizatorilor să navigheze meniul restaurantului și să plaseze comenzi.
* Transmite comenzile către bucătărie pentru pregătire.
* Permite chelnerilor să vizualizeze și să gestioneze comenzi.
* Cerințe nonfuncționale:
* Fiabilitate ridicată pentru a evita pierderea comenzilor.
* Performanță rapidă pentru a permite plasarea rapidă a comenzilor.
* Interfață ușor de utilizat pentru clienți și chelneri.

Structuri de date interne:

* Comenzi: Structura care conține informații despre fiecare comandă, inclusiv detaliile meniului, starea comenzii și informațiile despre client.

Constrângeri:

* Constrângeri de timp: Comenzile trebuie să fie procesate într-un timp limitat pentru a evita întârzierile.
* Constrângeri de stocare: Datele despre comenzile anterioare trebuie să fie stocate pentru referință și analiză ulterioară.

Compoziție:

* Serviciul SSGPCR este compus din module precum:
* Interfața cu utilizatorul pentru aplicația mobilă.
* Modulul de gestionare a comenzilor pentru chelneri.
* Modulul de comunicare cu bucătăria pentru transmiterea comenzilor.

Utilizatori/Interacțiuni:

* Clienții interacționează cu serviciul pentru a plasa comenzi.
* Chelnerii interacționează cu serviciul pentru a gestiona și monitoriza comenzile.

Procesare:

* Serviciul procesează comenzile primite de la clienți și le transmite către bucătărie pentru pregătire.
* Monitorizează starea comenzilor și actualizează interfața pentru clienți și chelneri în consecință.

Interfețe/Exporturi:

* Interfațe cu utilizatorul: Interfață intuitivă pentru plasarea comenzilor și gestionarea acestora.
* Interfață cu bucătăria: Interfață pentru transmiterea comenzilor și primirea actualizărilor despre starea comenzilor.

Raportare Design și Integrare:

* Volumul de date și traficul sunt monitorizate pentru a evalua performanța și pentru a face optimizări în funcție de nevoile utilizatorilor și ale restaurantului.