

**Дисциплина**

Приложно програмиране със C# и Visual Studio.Net

**Курсов проект**

**Тема:**

„Хотел“

Преподавател

**Студент:**

Илиян Кирилов Костов гл.ас.д-р Димитричка Николаева

фак.№ 23651224

Специалност : СИ

**Съдържание**

[**1.** **Задание** 3](#_Toc151928899)

[**2.** **Анотация** 5](#_Toc151928900)

[**3.** **Увод** 6](#_Toc151928901)

[**4.** **Обзор на съществуващите решения** 7](#_Toc151928902)

[**4.1** **Изводи** 7](#_Toc151928903)

[**4.2** **Цел и задачи** 7](#_Toc151928904)

[**5.** **Проектиране и описание на предлаганото решение** 8](#_Toc151928905)

[**5.1 Изисквания към програмна система** 8](#_Toc151928906)

[**5.2 Логически модел на програмната система** 8](#_Toc151928907)

[**5.3 Архитектура на системата** 9](#_Toc151928908)

[**5.4 Организация на данните** 10](#_Toc151928909)

[**5.5 Избор на език и среда на програмиране** 10](#_Toc151928910)

[**5.6 Реализация на програмната система** 11](#_Toc151928911)

[**5.6.1 Структура на данните** 19](#_Toc151928912)

[**5.6.2 Описание на програмните модули** 20](#_Toc151928913)

[**5.6.3 Структура и организация на потребителския интерфейс** 21](#_Toc151928914)

[**5.7 Формат на входните документи** 23](#_Toc151928915)

[**5.8 Формат на извеждащите справки** 24](#_Toc151928916)

[**5.9 Инструкции за работа с програмната система** 25](#_Toc151928917)

[**5.9.1 Ръководство на потребителя** 25](#_Toc151928918)

[**5.9.2 Инструкции и изисквания при инсталиране на системата** 28](#_Toc151928919)

[**5.9.3 Инструкции за поддържане на системата** 30](#_Toc151928920)

[**5.9.4 Изисквания към апаратното осигуряване** 30](#_Toc151928921)

[**4.** **Резултати от тестване на системата** 31](#_Toc151928922)

[**5.** **Използвана литература** 34](#_Toc151928923)

# **Задание**

Създайте База данни съдържаща информация необходима за хотелски резервации.

1) БД да съдържа следните таблици:

Клиенти

 идентификационен номер на клиент – числов ключ

 ЕГН - текст

 име - текст

 адрес - текст

 лична карта – текст

Стая

 номер на стая – числов тип

 категория /апартамент, единична, двойна, тройна/- текст

 етаж – числов тип

 брой легла – числов тип

 цена за наем на 1 легло /цената е фиксирана, независимо от категорията/ - double

Регистрация

 регистрационен номер – числов ключ

 дата на създаване на документа – тип дата

 идентификационен номер на клиент – числов тип

 номер на стая – числов тип

 начална дата за резервация – тип дата

 крайна дата за резервация – тип дата

2) Създайте форми за попълване на данни в съответните таблици (въвеждане, актуализиране

и изтриване на данни);

3) Създайте следни справки:

- всички свободни стаи към текущата дата и час, подредени по категорията на стаите;

- оборот от резервациите извлечен чрез задаване на период от време за:

 ден

 месец

 за последните три месеца

 за последните шест месеца

 година

- извеждане на всички имена, ЕГН на клиенти заели стаи през последната година.

- да се изчертае графика – най-често резервирана стая: Стая-Период по месеци.

4) Задължително реализацията да включва използване на класове, колекции – list, dictionary,

queue, stack и други структури съобразно тяхнат

# **Анотация**

Проекта се основава на управлението на резервации на клиенти в информационна система. Системата трябва да бъде визуално интуативна за администраторите на хотела, за да могат те лесно да се навигират към нейните и функционалности и изцяло да имат добро потребителско изживяване.

Основните функции на проекта поддържат създаване,актуализиране и изтриване на клиенти, стай и съответните им резервации.

Също така системата поддържа автентикация за влизане в приложението, заедно с SMS двукратна автентикация като допълнителен слой за защита.

Системата поддържа предварително определен набор от заявки към базата данни:

* Извеждане на всички свободни стай към текущ дата и час на администратора/потребителя
* Извеждане на оборот от регистрациите през определен от потребителя/администратора дати: ден,месец, последни 3/6 месеца, година
* Извеждане на потребителски данни, които са заели стои през последната година

Приложението е публично достъпен чрез достъп към домейна: <https://prod.d5izymva5csp3.amplifyapp.com/> . Хостването към приложението е изцяло сървърлес (serverless), с помоща на Amazon Web Services (AWS).

# **Увод**

Ръчното управление за резервирането на стаи в хотелите е сложна и повтаряща се задача - следенето на голяма бройка от клиенти и записа на всичките им потребителски и резервирани данни изисква прецизност от страна на администратора на хотела, както и обширно пространство от шкафове за файлове. Това води до значително административно натоварване и висока вероятност от допускане на грешки, което ще доведе до загуба на определена част от клиентелата на хотела.

Ето защо е необходима добра и надеждна система за хотелски резервации, която да улесни ежедневната работа на администраторите и едновременно да намали човешката грешка.

Един от алтернативните решения е разработка на приложение за резервации на хотели, което действа като централизирано пространство на управление на всички потребителски и хотелски резервации,както и да осигурява изчислителни услуги и организацията на данни.

Това позволява на хотелите значително да намалят броя на необходимите квалифицирани администратори.

По този начин, хотелите могат да насочат повече финансови ресурси към подобряване на клиентското обслужване и тяхното потребителско изживяване (experience), от момента на резервацията до връчването на ключовете при напускане.

# **Обзор на съществуващите решения**

1. **Booking.com**

Онлайн платформа, където хотели и други видове настаняване предоставят своите стаи на пътуващи гости от цял свят. Той също позволява на техните партньори за настаняване да управляват собствените си взаимоотношения с клиентите. Бизнес моделът предоставя своите услуги според „посреднически модел“ – цените на резервациите се определят изцяло от собствениците на имотите, а онлайн платформата взима комисионна от продажбите.

1. **Hotels.com**

Онлайн платформа, която поддържа мобилно приложение и уебсайт, предлагайки избор от хотели в различни дестинации.

Известна е с програмата си за награди, при която потребителите печелят безплатна нощувка за всеки 10 нощувки, които резервират чрез приложението. Той също така позволява на потребителите да преглеждат и променят резервациите си, да проследяват прогреса на своите награди и да получават достъп до специални сделки и отстъпки.

## **Изводи**

* Предимства:
  + междуплатформено приложение
  + независима платформа, давайки собствениците на имотите повече контрол към определяне на цената
  + поддържа програма за награди за лоялни клиенти
  + приложението поддържа различни езици
* Недостатъци:
  + голяма бройка от избори, може да претовари потребителите от оферти
  + филтрирането на хотели не е изцяло персонализирано

## **Цел и задачи**

Създаване на междуплатформено приложение, като предлага персонализирано средна бройка от имотни избори, с поддръжката на няколко различни езици. Приложението да поддържа програма за лоялни клиенти и да предлага разширено филтриране на хотели.

# **Проектиране и описание на предлаганото решение**

## **5.1 Изисквания към програмна система**

Целта на проекта е да се разработи междуплатформено приложение за управление на хотелски резервации, да подпомогне на администрирането на клиентските резервации, както и създаване на справки, за ефективно наблюдение над хотелските резервации.

## **5.2 Логически модел на програмната система**

A group of black circles

Description automatically generated

## **5.3 Архитектура на системата**

A diagram of a person

Description automatically generated

## **5.4 Организация на данните**

A screenshot of a computer program

Description automatically generated

## **5.5 Избор на език и среда на програмиране**

При проектиране на уеб приложение за резервиране на стай за хотели, особено ако приложението трябва да бъде международно, следните изисквания трябва да се удовлетворят, според сензитивността и мащабураемостта на приложението:

1. **Хостване: Amplify Hosting/AWS CloudFront**

* публично достъпен, като е интегриран с git workflow за хостване на fullstack сървърлес уеб приложение с продължително внедряване.

1. **Сигурност**

* прилагане на ( HTTPS ) за криптиране на данни в покой.

1. **Управление на бази данни: AWS DynamoDB**

- избрана система от бази данни (NoSQL)

**4. Управление на идентичността: AWS Cognito**

- система за идентификация на потребителите към системата

- управление на правата на достъп за индивидуални потребители и групи, като използва LDAP протокола

**5. Продължително внедряване на приложението**

- автоматизиране в целия жизнен цикъл на софтуерното разработване ( Software Development Life Cycle, SDLC)

Необходимите програмни езици за имплементирането на уеб-приложение са:

**1. C# .NET 6.0**

- бакенд логиката за валидиране на потребителските входни данни и записи в базата данни

**2. ReactJS 18.6.0 , ЕS7**

- фронтенда за визуализация на крайното приложение и извършване на API извиквания за Amazon бакенд услугите

**3. GraphQL**

- език за заявка към базата данни от различни източници от данни

- поддръжка на офлайн режим към приложението, като използва *Storage Engine*

## **5.6 Реализация на програмната система**

**Amplify.yml**

version: 1

backend:

  phases:

    build:

      commands:

        # Install .NET 6 SDK

        - curl -sSL https://dot.net/v1/dotnet-install.sh > dotnet-install.sh

        - chmod +x \*.sh

        - ./dotnet-install.sh -c 6.0 -InstallDir ./dotnet6

        - ./dotnet6/dotnet --version

        - export PATH="$(pwd)/dotnet6:$PATH"

        - echo $PATH

        # Install Amazon.Lambda.TestTool-6.0 global tool

        - ./dotnet6/dotnet tool install -g Amazon.Lambda.TestTool-6.0

        - "# Execute Amplify CLI with the helper script"

        - amplifyPush --simple

frontend:

  phases:

    preBuild:

      commands:

        - npm ci

    build:

      commands:

        - npm run build

  artifacts:

    baseDirectory: dist

    files:

      - "\*\*/\*"

  cache:

    paths:

      - node\_modules/\*\*/\*

**Home\_layout.jsx**

import FigHome from "../../ui-components/FigHome";

import FigCreateRoom from "../../ui-components/FigCreateRoom";

import FigUpdateUser from "../../ui-components/FigUpdateUser";

import FigUpdateRegistration from "../../ui-components/FigUpdateRegistration";

import FigUpdateRoom from "../../ui-components/FigUpdateRoom";

import FigCreateRegistration from "../../ui-components/FigCreateRegistration";

import FigBlank from "../../ui-components/FigBlank";

import { FuncHomeOverride } from "../overrides/home\_override";

import { View } from "@aws-amplify/ui-react";

import { FuncCreateRoomOverride } from "../overrides/create\_room\_override";

import {

  ComponentStateContext,

  HelpersContext,

} from "../../contexts/data\_models/context";

import React from "react";

import { AlertBoxHome } from "../../classes/components/alertBoxHome";

import { FuncCreateRegistrationOverride } from "../overrides/create\_registration\_override";

import { FuncUpdateClientOverride } from "../overrides/update\_client\_override";

import { FuncUpdateRoomOverride } from "../overrides/update\_room\_override";

import { FuncOutputTableOverride } from "../overrides/output\_table\_override";

import { OutputTableHome } from "../../classes/components/outputTableHome";

import { OutputTableTurnOver } from "../../classes/components/outputTableTurnOver";

import { FuncUpdateRegistrationOverride } from "../overrides/update\_registration\_override";

export default function HomeLayout() {

  const { ComponentStateObject } = React.useContext(ComponentStateContext);

  const { homeOverride } = FuncHomeOverride();

  // Create

  const { createRoomOverride } = FuncCreateRoomOverride();

  const { createRegistrationOverride } = FuncCreateRegistrationOverride();

  // Update

  const { updateClientOOverride } = FuncUpdateClientOverride();

  const { updateRoomOverride } = FuncUpdateRoomOverride();

  const { updateRegistrationOOverride } = FuncUpdateRegistrationOverride();

  // Output table

  const { outputTableOverride } = FuncOutputTableOverride();

  const { UtilsObject } = React.useContext(HelpersContext);

  return (

    <View position="relative" display="inline-block">

      {UtilsObject.showAlertBox && <AlertBoxHome />}

      <FigHome overrides={homeOverride} />

      {ComponentStateObject.showCreateRoomPage && (

        <View position="absolute" display="block" top="0">

          <FigCreateRoom overrides={createRoomOverride} />

        </View>)}

      {ComponentStateObject.showCreateRegistrationPage && (

        <View position="absolute" display="block" top="0">

          <FigCreateRegistration overrides={createRegistrationOverride} />

        </View>

      )}

      {ComponentStateObject.showUpdateClientPage && (

        <View position="absolute" display="block" top="0">

          <FigUpdateUser overrides={updateClientOOverride} />

        </View>

      )}

      {ComponentStateObject.showUpdateRoomPage && (

        <View position="absolute" display="block" top="0">

          <FigUpdateRoom overrides={updateRoomOverride} />

        </View>

      )}

      {ComponentStateObject.showUpdateRegistrationPage && (

        <View position="absolute" display="block" top="0">

          <FigUpdateRegistration overrides={updateRegistrationOOverride} />

        </View>

      )}

      {(ComponentStateObject.showAvailableRooms ||

        ComponentStateObject.showAllSSNs) && (

        <View position="absolute" display="block" top="0">

          <FigBlank overrides={outputTableOverride} />

          <View

            position="absolute"

            display="block"

            top="0"

            justifyContent="center"

            right="700px">

            <OutputTableHome />

          </View>

        </View>

      )}

      {ComponentStateObject.showBookingTurnover && (

        <View position="absolute" display="block" top="0">

          <FigBlank overrides={outputTableOverride} />

          <View

            position="absolute"

            display="block"

            top="0"

            justifyContent="center"

            right="700px">

            <OutputTableTurnOver />

          </View>

        </View>)}

    </View>);}

**App.js**

import "@aws-amplify/ui-react/styles.css";

import "./App.css";

import { Authenticator } from "@aws-amplify/ui-react";

import { BrowserRouter, Route, Routes } from "react-router-dom";

import HomeLayout from "./pages/layouts/home\_layout";

import { ComponentStateProvider } from "./providers/component\_provider/provider";

import { DataModelProvider } from "./providers/data\_models/provider";

import { HelpersProvider } from "./providers/helpers\_provider/provider";

const signUpFields = {

  signUp: {

    "custom:Passport": {

      placeHolder: "Enter your Passport Number",

      isRequired: true,

      label: "Passport",

      order: 4,},

    "custom:SocialSecurityNumber": {

      placeHolder: "Enter your Social Security Number",

      isRequired: true,

      label: "Social Security Number",

      order: 4,

    },

    address: {

      placeHolder: "Enter your Address",

      isRequired: true,

      label: "Address",

      order: 4,

    },

  },

};

export default function App() {

  return (

    <Authenticator formFields={signUpFields}>

      <BrowserRouter>

        <HelpersProvider>

          <DataModelProvider>

            <ComponentStateProvider>

              <Routes>

                <Route path="/" element={<HomeLayout />} />

              </Routes>

            </ComponentStateProvider>

          </DataModelProvider>

        </HelpersProvider>

      </BrowserRouter>

    </Authenticator>

  );

}

//export default withAuthenticator(App, components.Authenticator);

**Client.jsx**

import React, { useEffect, useState } from "react";

import { fetchUserAttributes } from "aws-amplify/auth";

import { HelpersContext } from "../../contexts/data\_models/context";

import { createClient } from "../../graphql/mutations";

import { getClient, listClients } from "../../graphql/queries";

export function Client() {

  const [cID, setCID] = useState("");

  const [name, setName] = useState([]);

  const [address, setAddress] = useState([]);

  const [ssn, setSsn] = useState([]);

  const [passport, setPassport] = useState([]);

  const [allClientIDNames, setAllClientIDNames] = useState({});

  const [selectedRegistrationName, setSelectedRegistrationName] = useState("");

  const { UtilsObject } = React.useContext(HelpersContext);

  const { logger, client } = UtilsObject;

  // Fetch user attributes

  useEffect(() => {

    async function fetchData() {

      try {

        const userAttributes = await fetchUserAttributes();

        setName(userAttributes.name);

        setAddress(userAttributes.address);

        setSsn(userAttributes["custom:SocialSecurityNumber"]);

        setPassport(userAttributes["custom:Passport"]);

        // Fetch all clients with the specified SSN

        const response = await client.graphql({

          query: listClients,

          variables: {

            filter: {

              ssn: { eq: userAttributes["custom:SocialSecurityNumber"] },

            },

          },

        });

        if (response.data.listClients.items.length === 0) {

          // No existing clients found, create a new client

          logger.info(

            "No existing clients found, creating a new client. Creating new client..."

          );

          await client.graphql({

            query: createClient,

            variables: {

              input: {

                name: userAttributes.name,

                address: userAttributes.address,

                ssn: userAttributes["custom:SocialSecurityNumber"],

                passport: userAttributes["custom:Passport"],

              },

            },

          });

        } //else logger.info("Existing client found, not creating a new client.");

      } catch (error) {

        console.error("Error in fetching/creating client:", error);

      }

    }

    fetchData();

  }, []);

  useEffect(() => {

    async function getAllCLients() {

      try {

        const response = await client.graphql({

          query: listClients,});

        response.data.listClients.items.forEach((client) => {

          setAllClientIDNames((prevState) => ({

            ...prevState,

            [client.id]: client.name,

          }));

        });

      } catch (error) {

        logger.error(error);}}

    getAllCLients();

  }, [client, logger]);

  const handleNameChange = (event) => {

    setName(event.target.value);

  };

  const handleAddressChange = (event) => {

    setAddress(event.target.value);

  };

  const handleSsnChange = (event) => {

    setSsn(event.target.value);

  };

  const handlePassportChange = (event) => {

    setPassport(event.target.value);

  };

  const setCllientByAllClientIDNames = async (

    event

    //registrationIDNamesArr

  ) => {

    const clientID = UtilsObject.dictFindKeyByValue(

      allClientIDNames,

      event.target.value

    );

    // Get a specific item

    const selectedClient = await client.graphql({

      query: getClient,

      variables: { id: clientID },

    });

    setCID(selectedClient.data.getClient.id);

    setName(selectedClient.data.getClient.name);

    setAddress(selectedClient.data.getClient.address);

    setSsn(selectedClient.data.getClient.ssn);

    setPassport(selectedClient.data.getClient.passport);

  };

  const handleSelectedRegistrationName = (event) => {

    setSelectedRegistrationName(event.target.value);

  };

  return {

    cID,

    setCID,

    selectedRegistrationName,

    handleSelectedRegistrationName,

    setCllientByAllClientIDNames,

    name,

    address,

    ssn,

    passport,

    handleNameChange,

    handleAddressChange,

    handleSsnChange,

    handlePassportChange,

    allClientIDNames,

    setAllClientIDNames,

  };

}

**Create\_registration.jsx**

import React from "react";

import { RegistrationCreateComponent } from "../../classes/components/registrationCreateComponent";

import { DataModelContext } from "../../contexts/data\_models/context";

export function FuncCreateRegistrationOverride() {

  const {

    RoomObject,

    RegistrationObject,

    isSubmitButtonLoading,

    ClientObject,

  } = React.useContext(DataModelContext);

  const { handleCancelClick, handleSubmitClick } =

    RegistrationCreateComponent();

  const isRegisterAttributesEmpty =

    RegistrationObject.dateEnd === "" ||

    RegistrationObject.dateStart === "" ||

    new Date(RegistrationObject.dateEnd).getTime() <=

      new Date(RegistrationObject.dateStart).getTime();

  const createRegistrationOverride = {

    select\_field\_client: {

      isRequired: true,

      value: RegistrationObject.selectedClientName,

      options: Object.values(ClientObject.allClientIDNames),

      errorMessage: "Client must not be empty!",

      onChange: (event) =>

        RegistrationObject.handleSelectedClientNameChange(event),

    },

    select\_field\_room: {

      isRequired: true,

      value: RegistrationObject.selectedRoomNumber,

      onChange: (event) =>

        RegistrationObject.handleSelectedRoomNumberChange(event),

      options: Object.values(RoomObject.allRoomsIDNumbers),

    },

    textfield\_datestart: {

      isRequired: true,

      hasError: RegistrationObject.dateStart === "",

      value: RegistrationObject.dateStart,

      type: "date",

      errorMessage: "Date must not be empty!",

      onChange: (event) => RegistrationObject.handleDateStartChange(event),

    },

    textfield\_dateend: {

      isRequired: true,

      hasError: RegistrationObject.dateEnd === "",

      value: RegistrationObject.dateEnd,

      type: "date",

      errorMessage: "Date must not be empty!",

      onChange: (event) => RegistrationObject.handleDateEndChange(event),

    },

    button\_submit: {

      onClick: (event) =>

        handleSubmitClick(RegistrationObject.selectedRegistrationIndex),

      isDisabled: isRegisterAttributesEmpty,

      isLoading: isSubmitButtonLoading,

    },

    button\_cancel: {

      onClick: (event) => handleCancelClick(event),

    },

  };

  return { createRegistrationOverride };

}

## **5.6.1 Структура на данните**

type Registration @model @auth(rules: [{allow: public}]) {

  id: ID!

  dateCreation: AWSDate

  dateStart: AWSDate

  dateEnd: AWSDate

  PKClient: ID! @index(name: "byClient")

  PKRoom: ID! @index(name: "byRoom")

}

type Room @model @auth(rules: [{allow: public}]) {

  id: ID!

  roomNumber: Int

  category: String

  floor: Int

  beds: Int

  price: Float

  FKRegistrations: [Registration] @hasMany(indexName: "byRoom", fields: ["id"])

}

type Client @model @auth(rules: [{allow: public}]) {

  id: ID!

  name: String

  ssn: String

  address: String

  passport: String

  FKRegistrations: [Registration] @hasMany(indexName: "byClient", fields: ["id"])

}

### **5.6.2 Описание на програмните модули**

A diagram of a diagram

Description automatically generated

### **5.6.3 Структура и организация на потребителския интерфейс**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a login form

Description automatically generatedA screenshot of a computer

Description automatically generated

## **5.7 Формат на входните документи**

**Форма за регистрация**

Формален контрол:

* Входни данни:
  + Лична карта
  + ЕГН
  + Адрес
  + Телефонен номер
  + Парола
  + Име

Логически контрол:

* Лична карта трябва да започва с 2 букви и останалите 7 цифри
* ЕГН трябва да бъде с дължина 10 и да бъде валидно ЕГН чрез контролна сума
* Парола трябва да бъде с дължина 8, да включва 8 малки и големи символа, да включва цифри и специални символи
* Телефонен номер трябва да бъде с дължина 10, да съдържа само цифри и да се потвърди чрез SMS съобщение

**Форма за създаване на регистрация**

Формален контрол:

* Входни данни:
  + Дата на създаване
  + Дата на свършване
  + Избор на клиент
  + Избор на стая

Логически контрол:

* Датата на свършване трябва да бъде преди датата на започване.
* Проверява се дали е избран клиент от полето
* Проверява се дали е избрана стая от полето

## **5.8 Формат на извеждащите справки**

| **room Number** | **category** | **floor** | **beds** | **price** | **Date Start** | **Date End** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Single | 1 | 1 | 1 | 2023-08-11 | 2023-08-01 |
| 4 | Triple | 3 | 3 | 2 | 2022-10-21 | 2022-10-22 |
| 4 | Triple | 3 | 3 | 2 | 2023-11-20 | 2023-11-25 |
| 4 | Triple | 3 | 3 | 2 | 2023-11-20 | 2023-11-01 |
| 5 | Triple | 8 | 8 | 30 | 2020-01-02 | 2023-01-01 |

TurnOver

DayMonthLast 3 monthsLast 6 monthsYear

Enter date

| **TurnOver** |
| --- |
| 2 |

| **Name** | **SSN** | **Room number** | **Date End** |
| --- | --- | --- | --- |
| Iliyan Kostov | 1111111111 | 4 | 2023-11-25 |
| Iliyan Kostov | 1111111111 | 4 | 2023-11-01 |
| Iliyan Kostov | 1111111111 | 1 | 2023-08-01 |
| Iliyan Kostov | 1111111111 | 5 | 2023-01-01 |

## **5.9 Инструкции за работа с програмната система**

### **5.9.1 Ръководство на потребителя**

Потребителски интерфейс

A screenshot of a computer

Description automatically generated

Главната страница включва главен екран за взаимодействие със приложението и горен тоолбар за избор на функиите:

* в ‘create’ полето на избора:
  + създаване на стая, регистрация
* в ‘update‘ полето на избора:
  + подновяване на потребители,стаи,регистрации
* в ‘inquery’ полето на избора:
  + извършване на следните заявки:
  + всички свободни стаи към текущата дата и час, подредени по категорията на стаите;
  + - оборот от резервациите извлечен чрез задаване на период от време за:
  +  ден
  +  месец
  +  за последните три месеца
  +  за последните шест месеца
  +  година
  + - извеждане на всички имена, ЕГН на клиенти заели стаи през последната година.
* Бутон `Logout` за изход от приложението

Създаване на регистрация интерфейс

A screenshot of a login form

Description automatically generated

Тази страница е за създаване на регистрация в приложението като преди създаването и е необходимо да се попълни всяко празно поле:

* `Date start`: стартова дата на заемане на стая във формат mm/dd/yyy
* `Date end`: завършваща дата на заемане на стая във формат mm/dd/yyy
* `Rooms`: поле на избор на всички в системата стаи във формат на номер на стая
* ‘Client’: поле на избор на всички в системата потребители
* ‘Submit’: потвърждение на регистрация с въведените данни
* `Cancel`: изход от текущия екран

Подновяване на потребител интерфейс

A screenshot of a computer

Description automatically generated

Тази страница е за подновяване на потребител в приложението като преди създаването и е необходимо да се попълни всяко празно поле:

* ` User`: поле на избор на всички в системата потребители
* ` Name`: текстово поле за име на потребителя
* ` Address`: текстово поле за адрес
* ‘Submit’: потвърждение на подновяване с въведените данни
* `Cancel`: изход от текущия екран

Подновяване на потребител интерфейс

A screenshot of a computer

Description automatically generated

Тази страница е за извеждане на всички имена, ЕГН на клиенти заели стаи през последната година.

Подновяване на потребител интерфейс

A screenshot of a computer

Description automatically generated

Тази страница е за оборот от резервациите извлечен чрез задаване на период от време за:

 ден

 месец

 за последните три месеца

 за последните шест месеца

 година

### **5.9.2 Инструкции и изисквания при инсталиране на системата**

Преди да започнете, уверете се, че сте инсталирали следното:

* Node.js v14.x или по-нова версия
* npm v6.14.4 или по-нова версия

- git v2.14.1 или по-нова версия

Ако все още нямате акаунт в AWS, ще трябва да създадете такъв.

Amplify (CLI) е необходим за създаване на AWS облачни услуги за приложение. Инсталиране на Amplify CLI.

npm install -g @aws-amplify/cli

За да настроите Amplify CLI на локалната машина, трябва да го конфигурирате да се свързва с вашия AWS акаунт.

Enter the access key of the newly created user:

? accessKeyId: # YOUR\_ACCESS\_KEY\_ID

? secretAccessKey: # YOUR\_SECRET\_ACCESS\_KEY

This would update/create the AWS Profile in your local machine

? Profile Name: # (default)

Successfully set up the new user.

За да започнете, първо създайте ново приложение React и след това инсталирайте и използвайте Amplify CLI, за да започнете да добавяте бакенд възможности към вашето приложение.

От директорията на вашите проекти изпълнете следната команда:

npm create vite@latest

✔ Project name: react-amplified

✔ Select a framework: › React

✔ Select a variant: › JavaScript

cd react-amplified

npm install

npm run dev

Сега, след като имате работещо приложение, е време да настроите Amplify, така че да можете да създадете необходимите бекенд услуги, необходими за поддръжка на приложението.

amplify init

? Enter a name for the project reactamplified

The following configuration will be applied:

?Project information

| Name: reactamplified

| Environment: dev

| Default editor: Visual Studio Code

| App type: javascript

| Javascript framework: react

| Source Directory Path: src

Пакетът aws-amplify е основната библиотека за работа с Amplify Libraries във вашите проекти:

npm install aws-amplify

След това конфигурирайте библиотеките на Amplify от страна на клиента, така че да може да взаимодейства с бекенд услугите.

Отворете src/main.jsx и добавете следния код под последното импортиране:

import { Amplify } from 'aws-amplify';

import amplifyconfig from './amplifyconfiguration.json';

Amplify.configure(amplifyconfig);

### **5.9.3 Инструкции за поддържане на системата**

За поддръжка на приложението, е препоръчително да се интегрира инструмент за автоматизация на софтуерните модули (dependencies), като Dependabot, за да може приложението да не въвежда никакви уязвимости от модулите. Също така ако се използва такъв dependency автоматизиран инструмент, който ще създава заявки за имплементиране на тези промени в сорс кода, още наричен Pull request, е добре да има имплементиран автоматизирано тестване (continuous integration), така че ще се осигури че самите подновявания на модули няма на повреди нормалната функционалност на приложението.

За по-нататъшно разработване на приложението, е препоръчително да се използва Git Flow стратегията със следване на заявки за имплементиране на промени в сорс кода.

За поддръжка на приложението в облака, да се използва Amplify Studio за визуално управление на бакенд средата и да се клонира N на брой инстанции на приложения, в зависимост от бройката на разработчиците. Така те могат те да си тестват новите имплементации, без да навредят продуктовото приложение.

### **5.9.4 Изисквания към апаратното осигуряване**

**- Процесор: >** Intel Core i5/i7/i9 or AMD Ryzen 5/7,

- **RAM**: > 8GB

- **Памет**: > 256 GB SSD

# **Резултати от тестване на системата**

**AuthHotelUsersProd**

> dotnet test.\TestAuth.csproj

  Determining projects to restore...

  All projects are up-to-date for restore.

  AuthHotelUsersProd -> C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\T

  u-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\AuthHote

  lUsersProd\src\bin\Debug\net6.0\AuthHotelUsersProd.dll

C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Cs

harp-Project-Hotel\amplify\backend\function\AuthHotelUsersProd\tests\TestAuth\

TestLambdaContext.cs(22,16): warning CS8618: Non - nullable property 'ClientCont

ext' must contain a non-null value when exiting constructor. Consider declarin

g the property as nullable. [C: \Users\ikostov2\Documents\VScode\_projects\Tu-Va

rna\Tu-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\AuthH

otelUsersProd\tests\TestAuth\TestAuth.csproj]

C: \Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Cs

harp-Project-Hotel\amplify\backend\function\AuthHotelUsersProd\tests\TestAuth\

TestLambdaContext.cs(22,16): warning CS8618: Non - nullable property 'Identity'

must contain a non-null value when exiting constructor. Consider declaring the

 property as nullable. [C: \Users\ikostov2\Documents\VScode\_projects\Tu-Varna\T

u-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\AuthHotelU

sersProd\tests\TestAuth\TestAuth.csproj]

C: \Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Cs

harp-Project-Hotel\amplify\backend\function\AuthHotelUsersProd\tests\TestAuth\

TestLambdaContext.cs(22,16): warning CS8618: Non - nullable property 'InvokedFun

ctionArn' must contain a non-null value when exiting constructor. Consider dec

laring the property as nullable. [C: \Users\ikostov2\Documents\VScode\_projects\

Tu-Varna\Tu-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\

AuthHotelUsersProd\tests\TestAuth\TestAuth.csproj]

TestAuth->C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Ma

  ster-Winter-Csharp-Project-Hotel\amplify\backend\function\AuthHotelUsersProd

  \tests\TestAuth\bin\Debug\net7.0\TestAuth.dll

Test run for C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\AuthHotelUsersProd\tests\TestAuth\bin\Debug\net7.0\TestAuth.dll (.NETCoreApp, Version = v7.0)

Microsoft (R) Test Execution Command Line Tool Version 17.7.2 (x64)

Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Passed!  - Failed:     0, Passed: 1, Skipped: 0, Total: 1, Duration: < 1 ms - TestAuth.dll(net7.0)

**TestCreateRoom**

oom > dotnet test.\TestCreateRoom.csproj

  Determining projects to restore...

  Restored C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Maste

  r-Winter-Csharp-Project-Hotel\amplify\backend\function\createHotelProd\src\c

  reateHotelProd.csproj (in 378 ms).

  1 of 2 projects are up-to-date for restore.

Test run for C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\createHotelProd\tests\TestCreateRoom\bin\Debug\net7.0\TestCreateRoom.dll (.NETCoreApp, Version = v7.0)

Microsoft (R) Test Execution Command Line Tool Version 17.7.2 (x64)

Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

[xUnit.net 00:00:02.91] createHotelProd.TestCreateRoom.TestRoomPayload[FAIL]

  Failed createHotelProd.TestCreateRoom.TestRoomPayload[721 ms]

  Error Message:

   Amazon.DynamoDBv2.AmazonDynamoDBException : The security token included in the request is invalid.

---- Amazon.Runtime.Internal.HttpErrorResponseException : Exception of type 'Amazon.Runtime.Internal.HttpErrorResponseException' was thrown.

   at Amazon.Runtime.AmazonServiceClient.Invoke[TResponse] (AmazonWebServiceRequest request, InvokeOptionsBase options)

   at Amazon.DynamoDBv2.AmazonDynamoDBClient.DescribeTable(DescribeTableRequest request)

   at Amazon.DynamoDBv2.DocumentModel.Table.DescribeTable(String tableName)

   at Amazon.Runtime.Internal.Util.Cache`2.GetValueHelper(TKey key, Boolean& isStaleItem, Func`2 creator)

   at Amazon.Runtime.Internal.Util.Cache`2.GetValue(TKey key, Func`2 creator, Boolean& isStaleItem)

   at Amazon.DynamoDBv2.DocumentModel.Table.LoadTableInfo(ICache`2 tableDescriptionCache)

   at Amazon.DynamoDBv2.DocumentModel.Table.LoadTable(IAmazonDynamoDB ddbClient, TableConfig config)

   at Amazon.DynamoDBv2.DocumentModel.Table.LoadTable(IAmazonDynamoDB ddbClient, String tableName, DynamoDBEntryConversion conversion, Boolean isEmptyStringValueEnabled)

   at Amazon.DynamoDBv2.DocumentModel.Table.LoadTable(IAmazonDynamoDB ddbClient, String tableName)

   at createHotelProd.AmazonDynamoDBService.GetItemAsync(String tableName, String keyName, String keyValue) in C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\createHotelProd\src\cDynamoDBItem.cs:line 24

   at createHotelProd.createHotelProd.LambdaHandler(LambdaEvent input, ILambdaContext context) in C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\createHotelProd\src\createHotelProd.cs:line 86

   at createHotelProd.TestCreateRoom.TestRoomPayload() in C:\Users\ikostov2\Documents\VScode\_projects\Tu-Varna\Tu-Varna-Master-Winter-Csharp-Project-Hotel\amplify\backend\function\createHotelProd\tests\TestCreateRoom\TestCreateRoom.cs:line 64

--- End of stack trace from previous location ---

----- Inner Stack Trace -----

   at Amazon.Runtime.HttpWebRequestMessage.GetResponse()

   at Amazon.Runtime.Internal.HttpHandler`1.InvokeSync(IExecutionContext executionContext)

   at Amazon.Runtime.Internal.PipelineHandler.InvokeSync(IExecutionContext executionContext)

   at Amazon.Runtime.Internal.Unmarshaller.InvokeSync(IExecutionContext executionContext)

   at Amazon.Runtime.Internal.PipelineHandler.InvokeSync(IExecutionContext executionContext)

   at Amazon.Runtime.Internal.ErrorHandler.InvokeSync(IExecutionContext executionContext)

Failed!  - Failed:     1, Passed: 0, Skipped: 0, Total: 1, Duration: < 1 ms - TestCreateRoom.dll(net7.0)

# **Използвана литература**

* Design and Implementation of Reservation Management System, Case Study: Grand Ville Hotels: <https://core.ac.uk/download/pdf/234677452.pdf>
* What is Booking.com? The complete guide: <https://www.littlehotelier.com/blog/running-your-property/online-booking-tips-from-booking-com-what-small-hotels-can-do-interview/>
* AWS Amplify Doc: <https://docs.amplify.aws/>
* AWS Amplify UI Doc: <https://ui.docs.amplify.aws/>