

**Дисциплина**

Управленски информационни системи

**Курсов проект**

**Тема:**

„Информационна система, подпомагаща продажбите в

магазин за строителни материали“

Преподавател

**Студент:**

Илиян Кирилов Костов гл.ас.д-р Диян Динев

фак.№ 23651224

Специалност : СИ

**Съдържание**

[**1.** **Задание** 3](#_Toc155459409)

[**2.** **Анотация** 4](#_Toc155459410)

[**3.** **Увод** 5](#_Toc155459411)

[**3.1** **Цел и задачи** 5](#_Toc155459412)

[**4.** **Проектиране и описание на предлаганото решение** 6](#_Toc155459413)

[**4.1 Изисквания към програмна система** 6](#_Toc155459414)

[**4.2 Логически модел на програмната система** 6](#_Toc155459415)

[**4.3 Архитектура на системата** 7](#_Toc155459416)

[**4.4 Организация на данните** 8](#_Toc155459417)

[**4.5 Избор на език и среда на програмиране** 8](#_Toc155459418)

[**4.6 Реализация на програмната система** 9](#_Toc155459419)

[**4.7 Структура на данните** 16](#_Toc155459420)

[**5 Структура и организация на потребителския интерфейс** 19](#_Toc155459421)

[**5.1 Прототип на потребителския интерфейс** 19](#_Toc155459422)

[**5.2 Формат на входните документи** 22](#_Toc155459423)

[**5.3 Формат на извеждащите справки** 23](#_Toc155459424)

[**5.4 Инструкции за работа с програмната система** 24](#_Toc155459425)

[**5.4.1 Ръководство на потребителя** 24](#_Toc155459426)

[**5.4.2 Инструкции и изисквания при инсталиране на системата** 28](#_Toc155459427)

[**5.4.3 Инструкции за поддържане на системата** 29](#_Toc155459428)

[**5.4.4 Изисквания към апаратното осигуряване** 30](#_Toc155459429)

[**6.** **Използвана литература** 31](#_Toc155459430)

# **Задание**

Да се създаде информационна система, подпомагаща продажбите в магазин за строителни материали. БД да съхранява информация за налични материали, клиенти, служители и направени продажби. В таблиците да се включат атрибути като номер на артикул, наименование, налично количество, вид на артикула, цена, продажба, артикул, количество клиент, име, телефон, фактура (да/не), мол, булстат, име на фирмата, служител, позиция, телефон и др.

Базата от данни трябва да е нормализирана. При създаване на таблиците изберете подходящ тип данни и други свойства на полетата. Да се валидират данните. Ограничете наличността да са само положително число.

Наименуванията на позициите да се избира от списък. Изберете правилни

ключови полета. Свържете таблиците с подходящи релации. В основните

таблици да има въведени минимум 10 записа коректни данни. Системата да

генерира справки на база информация съхранена в повече от една таблица.

Справките да са минимум 5. Например: Да се генерира справка, която да показва продадените артикули продадени от даден служител за определен период подредени по вид на артикула. Създадената информационна система да позволява въвеждане, корекция, актуализация и търсене на данни. Направените справки да могат да се експортират в подходящ файлов формат за отпечатване.

# **Анотация**

Проекта се основава на създаване на онлайн платформа за извършване на продажби в магазин за строителни материали. Системата трябва да бъде визуално интуитивна за служителите на фирмата, 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 други служители, клиенти, материали и съответните им продажби.

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

Системата поддържа предварително определен набор от заявки към базата данни:

* Извеждане на всички продажби от служител
* Извеждане на всички продажби от клиенти
* Извеждане на отчет на финансовата сума за всички продажби
* Извеждане на всички налични материали
* Извеждане на най-продавани материали

Приложението е публично достъпен чрез достъп към домейна: <https://prod.d2y1zehmljro8d.amplifyapp.com/> . Хостването към приложението е изцяло сървърлес (serverless), с помощта на Amazon Web Services (AWS).

# **Увод**

В съвременният свят, търговията със строителни материали се извършва или чрез доставчици, или директно от магазина, като в двата случая е на място. Всички тези транзакции между клиент–продавач трябва да бъдат записани от специализирана група от служители, за да може правилно да се проследят наличните материали в склада, да се оцени `търсенето и доставката` (supply and demand), управление на взаимоотношенията с клиенти и т.н.

При единичен магазин, създаването на отчет е сравнително контролируемо (дори само на хартиен формат), но при наличие на множество обекти, независимо дали се намират в една локация или в различни, самият процес става в пъти по-сложен, като същевременно увеличава вероятността на човешката грешка.

За тази цел идва необходимостта за създаване на информационна система, която да подпомогне на една фирма да създаде единна платформа за всички техни магазини, за улесняване на счетоводният процес и преодолявайки проблемите, свързани с физическото разстояние между тях.

Въвеждането на такава система в бизнесът им позволяват да насочат повече финансови ресурси към разширяване и развитието му, като откриване на нови магазини в различни места, без да влияе на нормалният счетоводен процес.

## **Цел и задачи**

Разработване на междуплатформено приложение за управление на продажбите и инвентара между клиенти и продавачи. Приложението трябва да бъде достатъчно стабилно, за да може издържи голям обем от потребителски трафик, идващ от различни страни.

# **Проектиране и описание на предлаганото решение**

## **4.1 Изисквания към програмна система**

Целта на проекта е да се разработи междуплатформено приложение за поддръжка на продажби за строителни материали, както и създаване на справки, за ефективно проследяване на наличните материали.

## **4.2 Логически модел на програмната система**

A diagram of a person

Description automatically generated

## **4.3 Архитектура на системата**

A diagram of a diagram

Description automatically generated

## **4.4 Организация на данните**

A screenshot of a computer

Description automatically generated

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

При проектиране на уеб приложение за продажби на строителни материали, особено ако приложението трябва да бъде многоезично, 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 транзит.

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. ReactJS 18.6.0 , ЕS7**

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

**2. GraphQL**

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

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

## **4.6 Реализация на програмната система**

**Amplify.yml**

version: 1

backend:

  phases:

    build:

      commands:

        - '# 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**

export default function HomeLayout() {

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

  const { createForm } = CreateFormComponent();

  const { updateForm } = UpdateFormComponent();

  const { homeOverride } = FuncHomeOverride();

  const { blankOverride } = BlankOverride();

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

  const handleSuccess = (message) => {

    UtilsObject.showAlertBoxFull("success", message, "success");

  };

  return (

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

      {UtilsObject.showAlertBox && <AlertBoxHome />}

      <FigHome overrides={homeOverride} />

      {

        /\* Create \*/

        (ComponentStateObject.showCreateMaterialPage ||

          ComponentStateObject.showCreateMaterialTypePage ||

          ComponentStateObject.showCreateCompanyPage ||

          ComponentStateObject.showCreateEmployeePage ||

          ComponentStateObject.showCreateCustomerPage ||

          ComponentStateObject.showCreateMallPage ||

          ComponentStateObject.showCreatePositionPage ||

          ComponentStateObject.showCreateSalesPage ||

          /\*Update \*/

          ComponentStateObject.showUpdateMaterialPage ||

          ComponentStateObject.showUpdateMaterialTypePage ||

          ComponentStateObject.showUpdateCompanyPage ||

          ComponentStateObject.showUpdateEmployeePage ||

          ComponentStateObject.showUpdateCustomerPage ||

          ComponentStateObject.showUpdateMallPage ||

          ComponentStateObject.showUpdatePositionPage ||

          ComponentStateObject.showUpdateSalesPage ||

          /\*Report \*/

          ComponentStateObject.reportSalesByEmployee ||

          ComponentStateObject.reportCustomerPurchaseOptions ||

          ComponentStateObject.reportFinancialSumReport ||

          ComponentStateObject.reportAvailableMaterials ||

          ComponentStateObject.reportPopularSoldMaterials) && (

          <View

            position="absolute"

            display="block"

            top="17%"

            zindex={1}

            justifyContent="center"

          >

            <FigBlank overrides={blankOverride} />

            <View

              position="absolute"

              display="block"

              top="0"

              justifyContent="center"

              right="820px"

            >

              {/\* Create form \*/}

              {createForm.map(

                ({ showPage, FormComponent, successMessage, setShowPage }) =>

                  showPage && (

                    <FormComponent

                      key={showPage}

                      onSuccess={() => {

                        handleSuccess(successMessage);

                        setShowPage(false);

                      }}/>)

              )}

              {/\*Update form \*/}

              {updateForm.map(

                ({

                  key,

                  showPage,

                  FormComponent,

                  onCancel,

                  onSuccess,

                  id,

                  overrides,

                }) =>

                  showPage && (

                    <FormComponent

                      key={key}

                      onCancel={onCancel}

                      onSuccess={onSuccess}

                      id={id}

                      overrides={overrides}

                    />)

              )}

              {/\*Report form \*/}

              <TableComponent />

            </View>

          </View>)

      }

    </Flex>

  );

}

**App.jsx**

export default function App() {

  const [positionIDNames, setPositionIDNames] = useState({});

  const [mallIDNames, setMallIDNames] = useState({});

  // Fetch position attributes

  useEffect(() => {

    async function fetchData() {

      try {

        const client = generateClient();

        // List all items

        const response = await client.graphql({

          query: listPositions,

        });

        const responseMall = await client.graphql({

          query: listMalls,

        });

        response.data.listPositions.items.forEach((position) => {

          setPositionIDNames((prevState) => ({

            ...prevState,

            [position.id]: position.name,

          }));

        });

        responseMall.data.listMalls.items.forEach((mall) => {

          setMallIDNames((prevState) => ({

            ...prevState,

            [mall.id]: mall.name,

          }));

        });

      } catch (error) {

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

    }

    fetchData();

  }, []);

  const signUpFields = {

    signUp: {

      name: {

        placeholder: "Enter your name",

        isRequired: true,

        label: "Name",

        order: 4,

      },

      phone\_number: {

        placeholder: "Enter your Phone Number",

        isRequired: true,

        label: "Phone",

        order: 4,

      },

    },

  };

  return (

    <Authenticator

      initialState="signUp"

      formFields={signUpFields}

      components={{

        SignUp: {

          FormFields() {

            const { validationErrors } = useAuthenticator();

            return (

              <>

                <SelectField

                  name="custom:custom:molAssigned"

                  label="Mol assigned"

                  options={Object.values(mallIDNames)}

                  errorMessage={validationErrors["custom:custom:molAssigned"]}

                />

                <SelectField

                  name="custom:custom:position"

                  label="Position"

                  options={Object.values(positionIDNames)}

                  errorMessage={validationErrors["custom:custom:position"]}

                />

                <Authenticator.SignUp.FormFields />

              </>

            );

          },

        },

      }}

    >

      <BrowserRouter>

        <HelpersProvider>

          <DataModelProvider>

            <ComponentStateProvider>

              <Routes>

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

              </Routes>

            </ComponentStateProvider>

          </DataModelProvider>

        </HelpersProvider>

      </BrowserRouter>

    </Authenticator>

  );

}

createFormComponent.jsx

export function CreateFormComponent() {

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

  const createForm = [

    {

      key: "Material",

      showPage: ComponentStateObject.showCreateMaterialPage,

      FormComponent: MaterialCreateForm,

      successMessage: "Material created successfully!",

      setShowPage: ComponentStateObject.setShowCreateMaterialPage,

    },

    {

      key: "MaterialType",

      showPage: ComponentStateObject.showCreateMaterialTypePage,

      FormComponent: MaterialTypeCreateForm,

      successMessage: "Material type created successfully!",

      setShowPage: ComponentStateObject.setShowCreateMaterialTypePage,

    },

    {

      key: "Customer",

      showPage: ComponentStateObject.showCreateCustomerPage,

      FormComponent: CustomerCreateForm,

      successMessage: "Customer created successfully!",

      setShowPage: ComponentStateObject.setShowCreateCustomerPage,

    },

    {

      key: "Employee",

      showPage: ComponentStateObject.showCreateEmployeePage,

      FormComponent: EmployeeCreateForm,

      successMessage: "Employee created successfully!",

      setShowPage: ComponentStateObject.setShowCreateEmployeePage,

    },

    {

      key: "Position",

      showPage: ComponentStateObject.showCreatePositionPage,

      FormComponent: PositionCreateForm,

      successMessage: "Position created successfully!",

      setShowPage: ComponentStateObject.setShowCreatePositionPage,

    },

    {

      key: "Mall",

      showPage: ComponentStateObject.showCreateMallPage,

      FormComponent: MallCreateForm,

      successMessage: "Mall created successfully!",

      setShowPage: ComponentStateObject.setShowCreateMallPage,

    },

    {

      key: "Sales",

      showPage: ComponentStateObject.showCreateSalesPage,

      FormComponent: SalesCreateForm,

      successMessage: "Sales created successfully!",

      setShowPage: ComponentStateObject.setShowCreateSalesPage,

    },

    {

      key: "Company",

      showPage: ComponentStateObject.showCreateCompanyPage,

      FormComponent: CompanyCreateForm,

      successMessage: "Company created successfully!",

      setShowPage: ComponentStateObject.setShowCreateCompanyPage,

    },

  ];

  return { createForm };

}

Home\_override.jsx

export function FuncHomeOverride() {

  const {

    selectFieldCreateUpdateOptions,

    selectFieldReport,

    handleSelectFieldUpdateOptions,

    handleSelectFieldInqueryOptions,

    handleSelectFieldReportOptions,

    handleSelectFieldCreateOptions,

    handleLogOutClick,

  } = HomeComponent();

  const { EmployeeObject } = React.useContext(DataModelContext);

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

  const homeOverride = {

    text\_page: {

      children: UtilsObject.homeText,

    },

    text\_name: {

      children: EmployeeObject.name,

    },

    select\_field\_create: {

      onChange: (event) => handleSelectFieldCreateOptions(event),

      style: { backgroundColor: "transparent", color: "transparent" },

      options: selectFieldCreateUpdateOptions,

    },

    select\_field\_update: {

      onChange: (event) => handleSelectFieldUpdateOptions(event),

      style: { backgroundColor: "transparent", color: "transparent" },

      options: selectFieldCreateUpdateOptions,

    },

    select\_field\_query: {

      onChange: (event) => handleSelectFieldInqueryOptions(event),

      style: {

        backgroundColor: "transparent",

        color: "transparent",

        // display: "none",

      },

      options: selectFieldCreateUpdateOptions,

    },

    select\_field\_report: {

      onChange: (event) => handleSelectFieldReportOptions(event),

      style: { backgroundColor: "transparent", color: "transparent" },

      options: selectFieldReport,

    },

    button\_logout: {

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

    },

  };

  return { homeOverride };

}

## **4.7 Структура на данните**

type MaterialType @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  name: String

  Materials: [Material] @hasMany(indexName: "byMaterialType", fields: ["id"])

}

type Company @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  name: String

  Customers: [Customer] @hasMany(indexName: "byCompany", fields: ["id"])

}

type Mall @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  name: String

  Employees: [Employee] @hasMany(indexName: "byMall", fields: ["id"])

}

type Position @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  name: String

  Employees: [Employee] @hasMany(indexName: "byPosition", fields: ["id"])

}

type Sales @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  quantitySold: Int

  salesDate: AWSDate

  totalPrice: Float

  employeeID: ID! @index(name: "byEmployee")

  customerID: ID! @index(name: "byCustomer")

  materialID: ID! @index(name: "byMaterial")

}

type Material @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  name: String

  quantityAvailable: Int

  price: Float

  Sales: [Sales] @hasMany(indexName: "byMaterial", fields: ["id"])

  materialtypeID: ID! @index(name: "byMaterialType")

}

type Customer @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  name: String

  phone: AWSPhone

  invoiceStatus: Boolean

  bulstat: String

  Sales: [Sales] @hasMany(indexName: "byCustomer", fields: ["id"])

  companyID: ID! @index(name: "byCompany")

}

type Employee @model @auth(rules: [{allow: public}, {allow: owner}]) {

  id: ID!

  name: String

  phone: AWSPhone

  Sales: [Sales] @hasMany(indexName: "byEmployee", fields: ["id"])

  positionID: ID! @index(name: "byPosition")

  mallID: ID! @index(name: "byMall")

}

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

## **5.1 Прототип на потребителския интерфейс**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

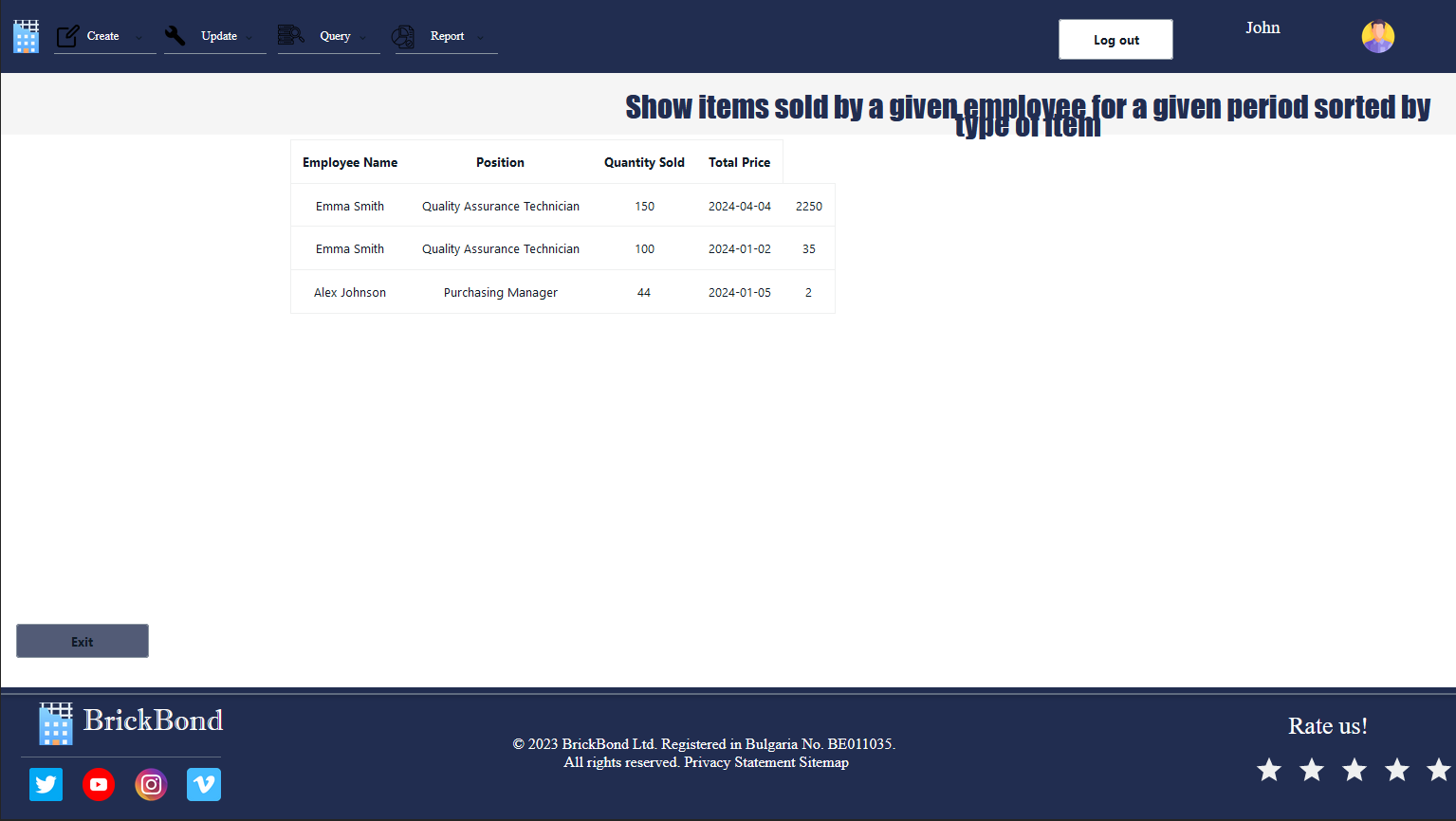
Description automatically generatedA screenshot of a computer

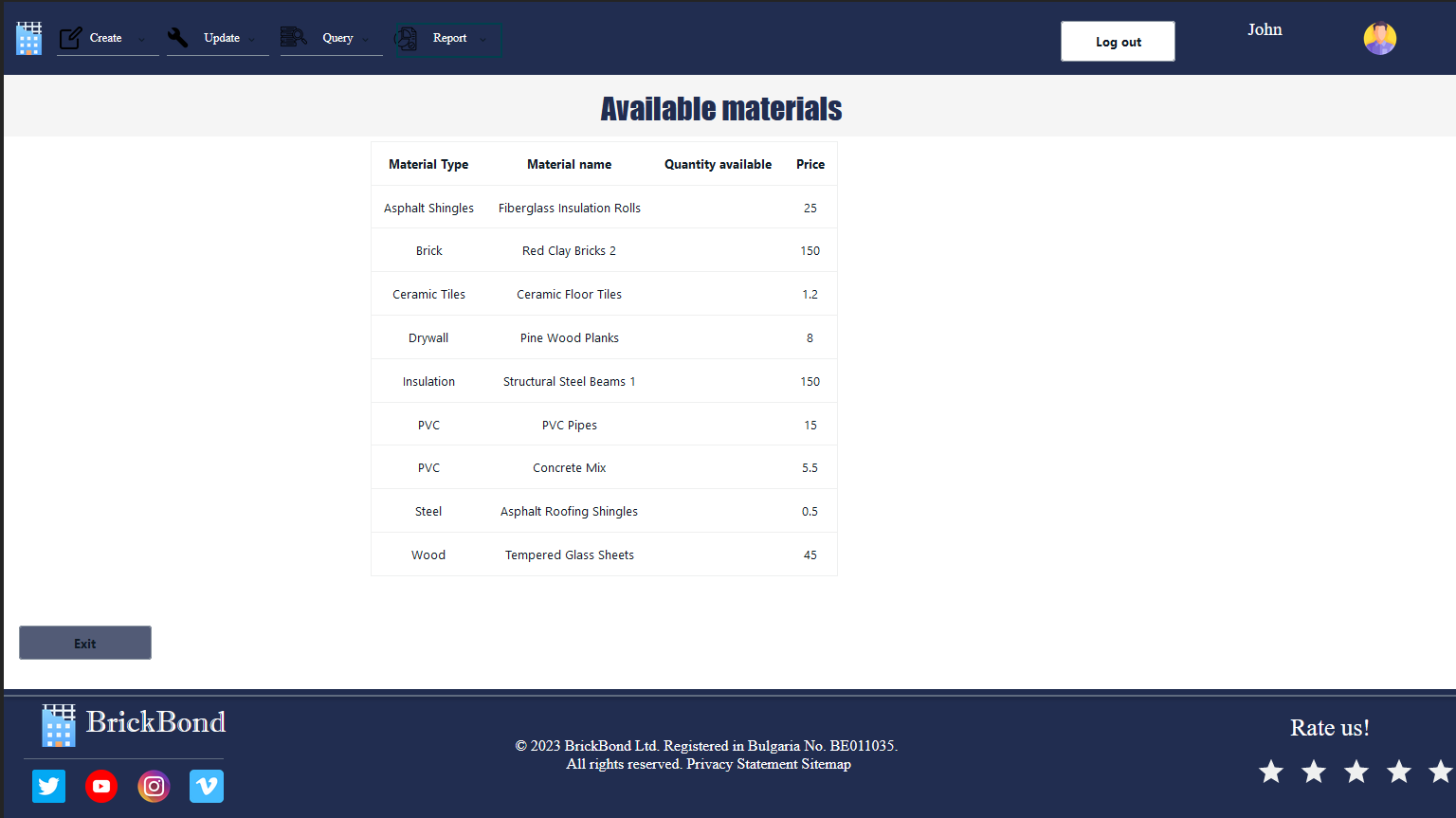
Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated





## **5.2 Формат на входните документи**

**Форма за регистрация**

Формален контрол:

* Входни данни:
  + Мол
  + Длъжност
  + Име
  + Телефонен номер
  + Имейл адрес
  + Парола

Логически контрол:

* Длъжността трябва да се избере от вече съществуващите, намиращи се в базата данни
* Мол трябва да се избере от вече съществуващите, намиращи се в базата данни
* 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

**Форма за създаване на материал**

Формален контрол:

* Входни данни:
  + Име
  + Количество
  + Цена
  + Вид материал

Логически контрол:

* Името не трябва да бъде празно
* Проверява се дали количеството е цяло число
* Проверява се дали цената е число
* Вида материал трябва да се избере от вече съществуващите, намиращи се в базата данни

## **5.3 Формат на извеждащите справки**

* Отчет за продажби по служител

| **Employee Name** | **Position** | **Quantity Sold** | **Total Price** |
| --- | --- | --- | --- |
| Emma Smith | Quality Assurance Technician | 150 | 2024-04-04 | 2250 |
| Emma Smith | Quality Assurance Technician | 100 | 2024-01-02 | 35 |
| Alex Johnson | Purchasing Manager | 44 | 2024-01-05 | 2 |

* История на покупките на клиента

| **Customer Name** | **Phone** | **Invoice status** | **Bulstat** | **Company** | **Sales date** | **Quantity Sold** | **Material name** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BrightFuture Developments |  |  | 321987654 | Fortress Foundations Corp3 | 2024-04-04 | 150 | Asphalt Roofing Shingles |
| BrightFuture Developments |  |  | 321987654 | Fortress Foundations Corp3 | 2024-01-02 | 100 | Pine Wood Planks |
| Pinnacle Building Co. |  |  | 432198765 | Skyline Building Solutions | 2024-01-05 | 44 | PVC Pipes |

* Отчет за финансовата сума

| **Total Sales** | **Number of transactions** | **Average Sales** |
| --- | --- | --- |
| 2287 | 3 | 762.3333333333334 |

* Налични материали

| **Material Type** | **Material name** | **Quantity available** | **Price** |
| --- | --- | --- | --- |
| Asphalt Shingles | Fiberglass Insulation Rolls |  | 25 |
| Brick | Red Clay Bricks 2 |  | 150 |
| Ceramic Tiles | Ceramic Floor Tiles |  | 1.2 |
| Drywall | Pine Wood Planks |  | 8 |
| Insulation | Structural Steel Beams 1 |  | 150 |
| PVC | PVC Pipes |  | 15 |
| PVC | Concrete Mix |  | 5.5 |
| Steel | Asphalt Roofing Shingles |  | 0.5 |
| Wood | Tempered Glass Sheets |  | 45 |

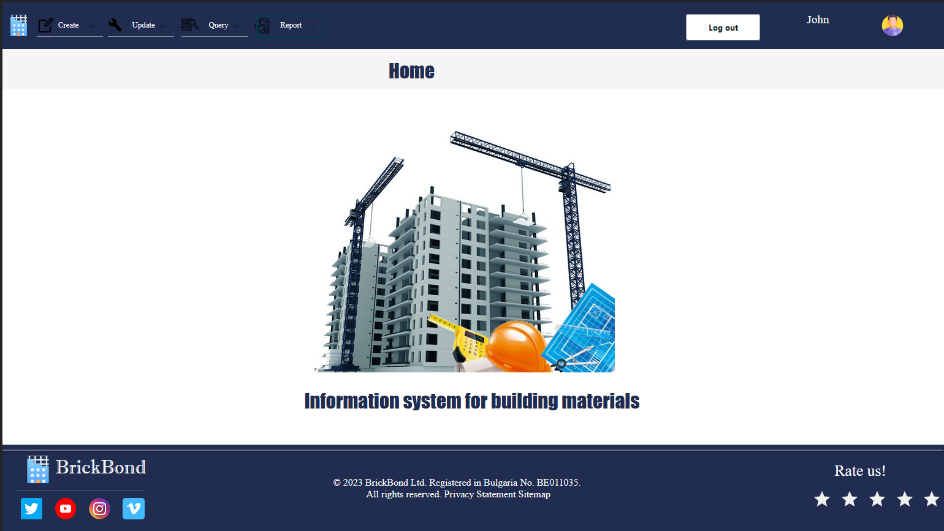
* Най-продавани материали

| **Material Type** | **Material name** | **Total quantity sold** | **Total Sales price** |
| --- | --- | --- | --- |
| Steel | Asphalt Roofing Shingles | 150 | 2250 |
| Drywall | Pine Wood Planks | 100 | 35 |
| PVC | PVC Pipes | 44 | 2 |

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

### **5.4.1 Ръководство на потребителя**

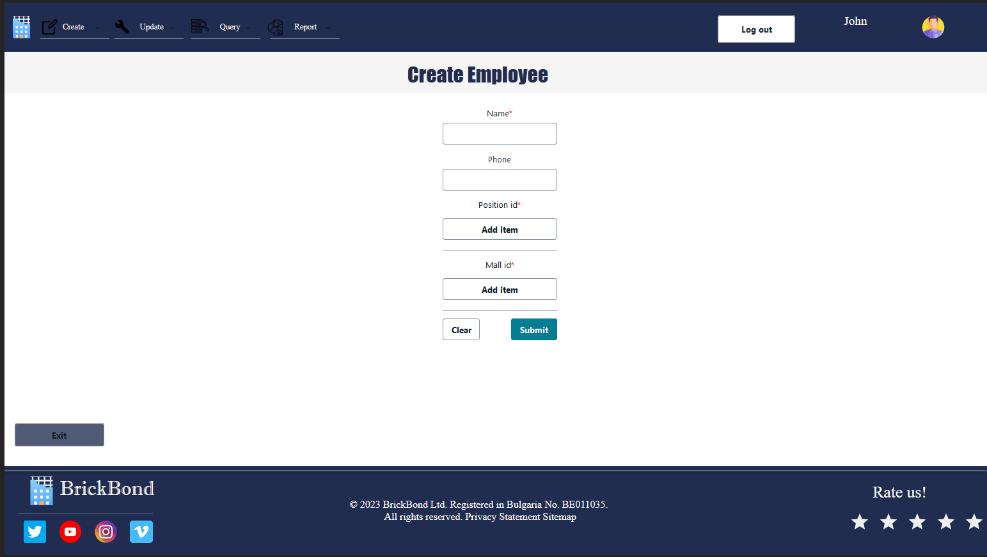
Потребителски интерфейс



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

* в ‘create’ полето на избора:
  + създаване на клиент, продажба, материал, вид материал, служител, длъжност, мол
* в ‘update‘ полето на избора:
  + подновяване на клиент,продажба,материал,вид материал, служител, длъжност, мол
* в ‘report` полето на избора:
  + извършване на следните заявки:
    - Отчет за финансовата сума
    - История на покупките на клиента
    - Отчет за продажби по служител
    - Налични материали
    - Най-продавани материали
* Бутон `Logout` за изход от приложението

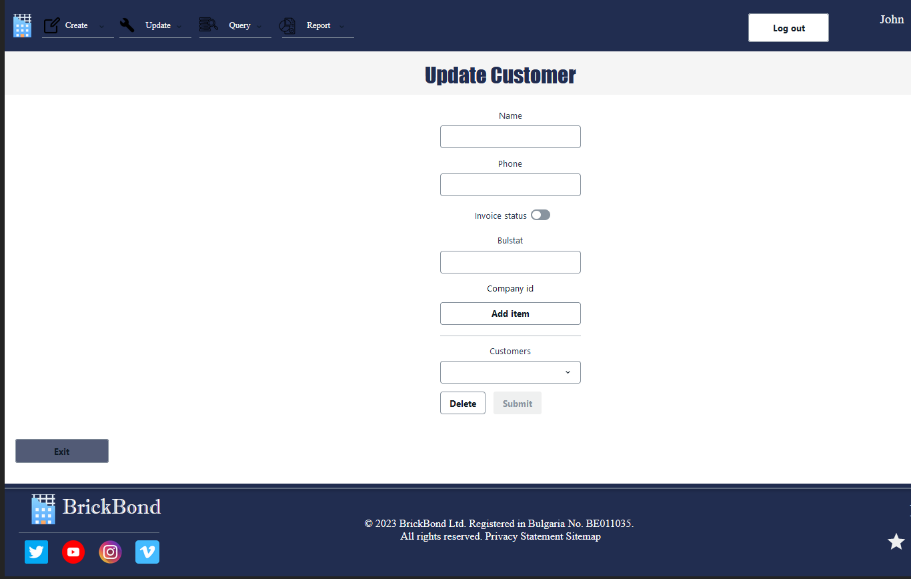
Създаване на служител интерфейс



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

* `Name `: Име на служителя
* `Phone `: Телефонен номер на служителя, който да спазва E.164 формат
* `Position id`: поле на избор на всички длъжности
* ‘Mall id`: поле на избор на всички молове
* ‘Submit’: потвърждение на регистрация с въведените данни
* `Cancel`: изход от текущия екран

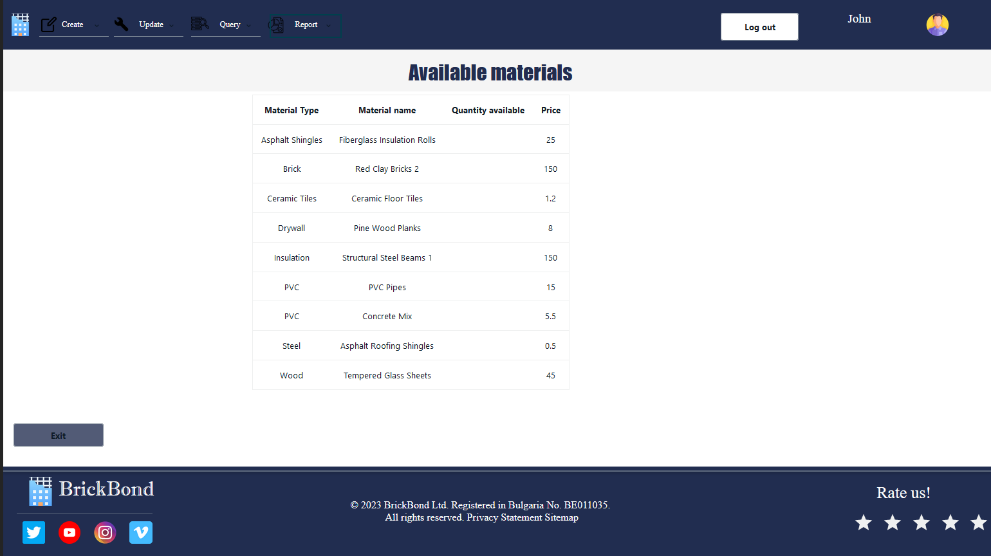
Подновяване на съществуващ клиент интерфейс



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

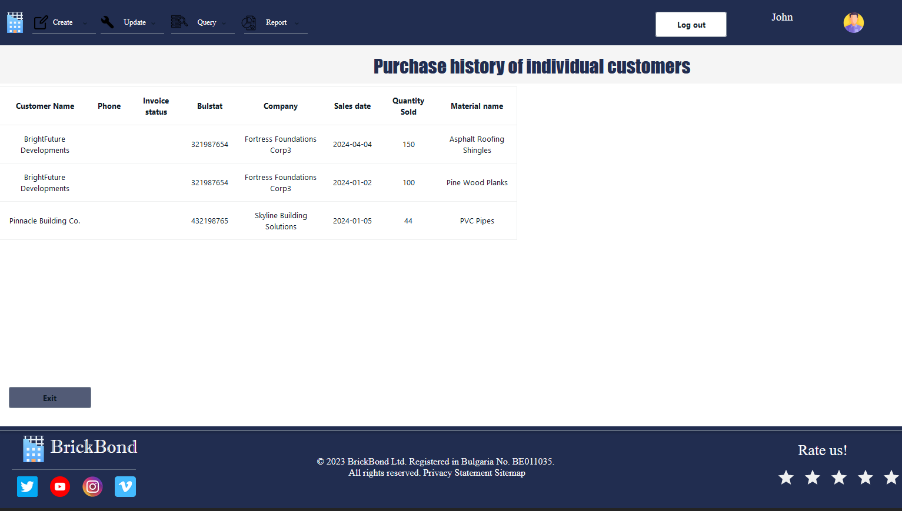
* ` Customers`: поле на избор на всички в системата клиенти
* ` Name`: текстово поле за името на клиента
* `Phone `: Телефонен номер на клиента, който да спазва E.164 формат
* `Invoice status`: Да/Не избор дали клиента е връчил фактура
* `Bulstat`: текстово поле за булстат на клиента
* ‘Company id`: поле на избор на всички компании
* ‘Submit’: потвърждение на подновяване с въведените данни
* `Cancel`: изход от текущия екран

Справка за всички налични материали



Тази страница е за извеждане на всички налични материали в склада, с информация за категорията материал, име, наличното количество и цена.

История на покупките на клиенти интерфейс



Тази страница е за извеждане на всички клиентови транзакции, съдържащи името на потребителя, тел. номера, статуса на фактурата, булстат, името на компанията, датата на извършената транзакция, продадено количество и името на продадения материал.

### **5.4.2 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);

### **5.4.3 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.

### **5.4.4 Изисквания към апаратното осигуряване**

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

- **RAM**: > 8GB

- **Памет**: > 256 GB SSD

# **Използвана литература**

* Web-based information system sales by Suwita, F & Sholihat, S & Dewi, N.: <https://www.researchgate.net/publication/349578493_Web-based_information_system_sales>
* AWS Amplify Doc: <https://docs.amplify.aws/>
* AWS Amplify UI Doc: <https://ui.docs.amplify.aws/>
* ReactJS Doc: <https://react.dev/>
* ViteJS Doc: <https://vitejs.dev/guide/>