**Ministerul Educației Și Cercetării Al Republicii Moldova**

**Colegiul „Iulia Hasdeu” Din Cahul**

**Catedra TIC**

**RAPORT**

**Practica Tehnologică**

**Specialitatea: 61310 Programarea Și Analiza Produselor Program**

**Calificarea: Asistent Programator**

**Semestrul VI**

**Grupa PAPP 2031**

**Efectuat:**



(Numele, prenumele studentului; grupa.)

**Verificat:**



(Numele, prenumele coordonatorului.)

**Nota:**



Cuprins

[**1. Condiția problemei. 3**](#_ljojmxtxadk1)

[**2. Conținutul aplicației. 3**](#_i3otjcl01565)

[A. Modelul de date. 3](#_a5c4w9mr0jb)

[B. Modelul interfeței cu utilizatorul. 3](#_46ge731h5ipo)

[**3. Proiectarea interfeței aplicației. 3**](#_2jhxk65wojil)

[A. Proiectul aplicației. 3](#_fn8g1hympt5r)

[B. Meniurile aplicației. 3](#_5m0y1v3sv7vf)

[C. Interfața aplicației. 4](#_jducfitcjzyn)

[**4. Elaborarea algoritmului aplicației. 5**](#_vn3czm7alq98)

[A. Proiectul aplicației. 5](#_rfaibveihpuk)

[B. Ferestrele aplicației. 6](#_rfdv3yef8u0w)

[**5. Funcționarea algoritmului de procesare a mesajelor. 6**](#_ezfa4n8ri88r)

[**6. Mentenanța aplicației. 6**](#_fpbw4ya4io3w)

[**7. Testarea aplicației. 7**](#_s6qge2kqhc1g)

[**8. Concluzie. 8**](#_fil7fqxmyfme)

[**9. Bibliografie. 9**](#_pp4y15q04dfg)

[**10. Anexă. 10**](#_64gb7953wp8s)

# Condiția problemei.

Crearea unui chatbot pentru o companie de asigurări. Informaţiile despre clienți și servicii sunt stocate într-o structură generică.

Elaboraţi un proiect, a cărei aplicaţie respectivă, folosind meniuri, clase şi subprograme, realizează, la solicitarea utilizatorului, următoarele funcţionalităţi:

1. Salută utilizatorul și oferă o prezentare a chatbot-ului.
2. Oferă informații despre tipurile de asigurări oferite de companie.
3. Răspunde la întrebări despre modalitățile de plată disponibile pentru asigurările oferite.
4. Furnizează informații despre beneficiile și limitările diferitelor tipuri de asigurări.
5. Asistă utilizatorul în alegerea celei mai bune opțiuni de asigurare în funcție de nevoile lor specifice.
6. Verifică disponibilitatea unei asigurări pentru o anumită perioadă de timp și furnizează un citat de preț.
7. Oferă instrucțiuni și asistență pentru a depune o cerere de despăgubire.
8. Oferă informații despre procedurile și termenele de plată a despăgubirilor.
9. Verifică starea unei cereri de despăgubire și oferă actualizări utilizatorului.
10. Furnizează numărul de telefon și adresa de e-mail a companiei de asigurări pentru a contacta un reprezentant al companiei.

# Conținutul aplicației.

## Modelul de date.

***Modelul datelor*** *- principalele structuri de date folosite, precum și schema bazei de date.*

Aplicația nu va folosi o bază de date, dar va stoca datele necesare în matrici de tipul „string”, unidimensionale și bidimensionale. Motivul pentru aceasta este că implementarea unei baze de date în aplicație ar fi problematică și lipsită de necesitate, din cauza condiției problemei și algoritmul creat pentru rezolvarea ei.

## Modelul interfeței cu utilizatorul.

***Modelul interfeţei cu utilizatorul*** *- interfaţa cu utilizatorul şi succesiunea ferestrelor acesteia.*

Interfața aplicației va fi de tip „dialog”, adică utilizatorul va introduce un mesaj într-o casetă text, iar răspunsul programului va fi scris pe o etichetă cu funcție de chat log.

# Proiectarea interfeței aplicației.

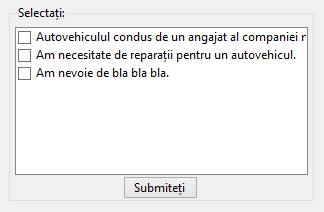
## Proiectul aplicației.

Aplicația va avea două funcționalități de bază:

* Procesarea întrebărilor utilizatorului;
* Determinarea răspunsului cel mai apropiat.

## Meniurile aplicației.

Aplicația are un singur meniu, constând dintr-o listă de checkboxuri și un buton. Acest meniu este utilizat pentru a genera cel mai apropiat răspuns pe baza unor criterii.



*figura 1.1.*

## Interfața aplicației.

Interfața aplicației este formată din două casete text, prima fiind the tip „rich text” și ocupând majoritatea din spațiul ferestrei, a doua fiind de tip text simplu, amplasată la marginea de jos a ferestrei aplicației, și un buton amplasat la dreapta casetei de tip text simplu.

Primul element va servi ca chat log, fiind unica metodă de comunicare a bot-ului cu utilizatorul, al doilea element va fi utilizat pentru scrierea întrebărilor utilizatorului, și al treilea element va trimite mesajul scris chatbot-ului pentru analizare.



*figura 1.2.1:*

1. *Chat log-ul.*
2. *Caseta de text.*
3. *Butonul de trimitere.*



*figura 1.2.2.*

1. *Group box-ul care conține lista checkbox și butonul de trimitere. Ascuns implicit.*
2. *Lista checkbox;*
3. *Butonul de trimitere;*
4. *Caseta de text și butonul de trimitere ale mesajelor au fost dezactivate.*

# Elaborarea algoritmului aplicației.

## Proiectul aplicației.

Algoritmul va avea ca bază clasa *ChatBot*, care conține matrici de date necesare pentru procesarea mesajelor trimise de către utilizator, și event handler-ul butonului *button\_Send*, care va procesa mesajul cu ajutorul matricelor de date și va afișa un răspuns în chat log.

Lista claselor, procedurilor și event handler-urilor aplicate în codul programului:

* *ChatBot* - clasa ce conține proprietățile și vocabularul chatbot-ului;
* *Name, Greeting* - șiruri de text utilizate generic în conversație;
* *Keywords* - matrice bidimensională de cuvinte folosite pentru înțelegerea mesajelor utilizatorului;
* *Answers* - matrice unidimensională de fraze folosite pentru a returna un răspuns utilizatorului;
* *BestInsuranceOption()* - procedură care analizează criteriile selectate în lista checkbox și trimite un răspuns;
* *BIO\_Options* - matrice unidimensională de fraze care servește ca sursă de date pentru lista checkbox;
* *BIO\_Answers* - matrice unidimensională de șiruri de caractere folosită în generarea automată al unui răspuns de către *BestInsuranceOption()*.
* *button\_Send\_Click()* - event handler care procesează mesajul utilizatorului utilizând cuvinte-cheie, determină cel mai potrivit răspuns predeterminat dintr-o listă și îl afișează în chat log-ul aplicației.

## Ferestrele aplicației.

| *figura 1.3.1.*  *Fereastra principală, mod implicit.* | *figura 1.3.2.*  *Fereastra principală după apelarea* BestInsuranceOption()*.* |
| --- | --- |

# Funcționarea algoritmului de procesare a mesajelor.

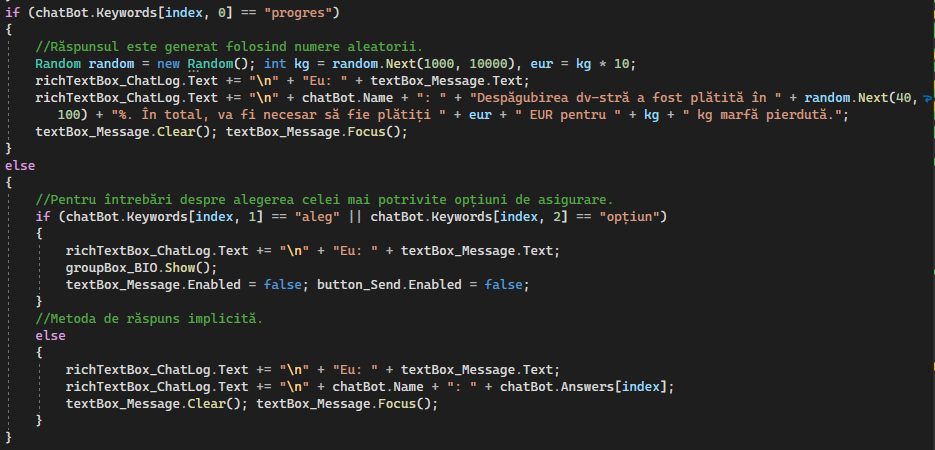
1. Programul caută instanțe de cuvinte-cheie preluate dintr-o matrice bidimensională, le numără și, pentru rândul cu cele mai multe cuvinte-cheie în comun cu mesajul, se păstrează valoarea indexului într-o variabilă numerică. (Anexa 1.1.)
2. Dacă rândul indexat se referă la o întrebare despre starea cererii de despăgubire, programul generează un răspuns automat utilizând numere aleatorii. (Anexa 1.2.)
3. Dacă rândul indexat se referă la o întrebare despre alegerea unei opțiuni de asigurare, programul apelează *BestInsuranceOption()* și generează un răspuns pe baza căror checkbox-uri au fost selectate. (Anexa 1.3.)
4. De altfel, programul preia un răspuns dintr-o matrice unidimensională cu indexul păstrat la pasul 1 și îl afișează pe chat log. (Anexa 1.4.)

# Mentenanța aplicației.

Mentenanța aplicației este un moment foarte important în dezvoltarea aplicațiilor de orice gen. Aceasta permite o organizare mai efectivă a muncii, îndeosebi când de program se ocupă mai mulți programatori. Câteva metode de mentenanță sunt:

* Comentarea secvențelor de cod;
* Structurarea programului;
* Folosirea metodelor de scriere de cod „clean code”.

Pentru mentenanța aplicației, în codul programului sunt prezente comentarii, care au scopul de a explica unele secvențe de cod (figura 1.4). Codul programului este asemenea structurat, pentru găsirea ușoară a secvențelor de cod.



*figura 1.4.*

*Exemplu de comentarii în codul programului.*

# Testarea aplicației.

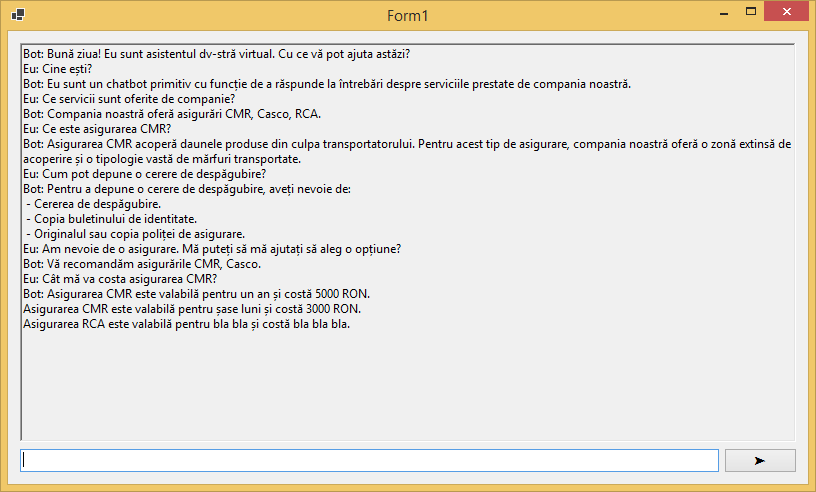
Pentru o experiență bună, utilizatorul nu trebuie să întâlnească erori în timpul lucrului cu aplicația. Găsirea și repararea acestora permite evitarea eventualelor probleme a lucrului aplicației.

Există diverse tehnici de căutare a erorilor, care sunt folosite și în aplicația dată:

* Structurarea bună a aplicației;
* Controlul datelor introduse cu blocurile „if”.

În aplicația dată, o mare parte a programului îl constituie introducerea de șiruri de caractere de către utilizator. O mare parte a programului este procesarea limbajului uman introdus prin text, de aceea este necesar ca programul să recunoască subiectul mesajului introdus de utilizator. Pentru aceasta, mai multe blocuri „if” și alte verificări sunt implementate ca programul să poată identifica subiectul mesajului cu ajutorul întregului său vocabular, returnând un răspuns de confuzie în cazul lipsei de cuvinte în comun. Programul nu are necesitate de blocuri „try-catch-finally” datorită acesteia.

Programul poate returna excepții în algoritmul de procesare al limbajului doar în cazul în care numărul de rânduri în matricea de vocabular nu este egal cu numărul de elemente în matricea de răspunsuri, deoarece programul depinde de utilizarea indecșilor echivalenți pentru amândouă array-uri.



*figura 1.5.*

*Testarea aplicației. Conversație între utilizator și chatbot.*

# Concluzie.

Scopul propus de această lucrare practică a fost de a dezvolta o aplicație pe baza *Windows Forms* și limbajului de programare *C#*. În cursul practicii, am avut de efectuat o listă de sarcini pentru a obține produsul final.

Pe parcursul elaborării am întâlnit dificultăți la elaborarea unui model de procesare al limbajului, informându-mă de la mai multe surse despre modul de funcționare al chatboților primitivi, găsind o sursă de inspirație pentru structurarea algoritmului de procesare. Sunt mulțumit de faptul că codul meu este structurat într-un mod rezonabil, dar sunt dezamăgit de inabilitatea mea de a elabora un algoritm de procesare al limbajului uman mai puternic, cu toate că metoda curentă este de ajuns. Dar sunt mulțumit de faptul că am avut șansa să îndeplinesc o sarcină cu care m-aș putea întâlni în viitorul meu profesional.

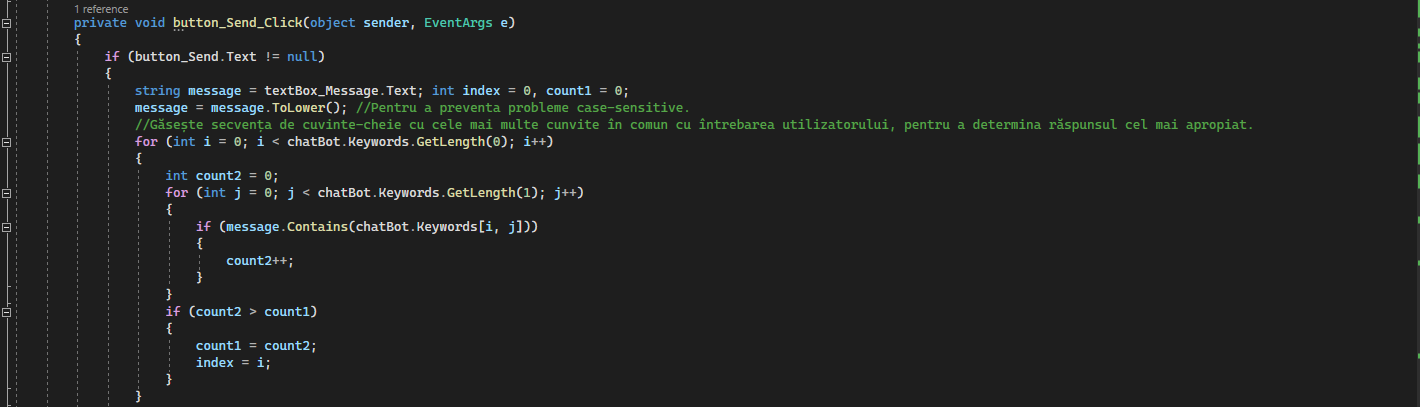
În general, practica individuală a avut un impact benefic asupra experienței noastre în domeniu, ceea ce ne va ajuta în continuare să obținem cunoștințe și succese profesionale.

# Bibliografie.

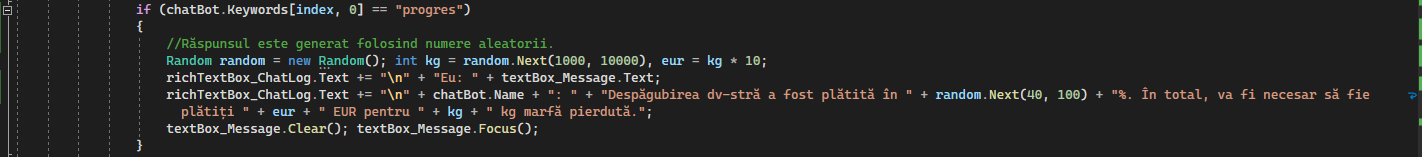
* [„The History Of Chatbots – From ELIZA to ChatGPT” (Onlim)](https://onlim.com/en/the-history-of-chatbots/)

# Anexă.

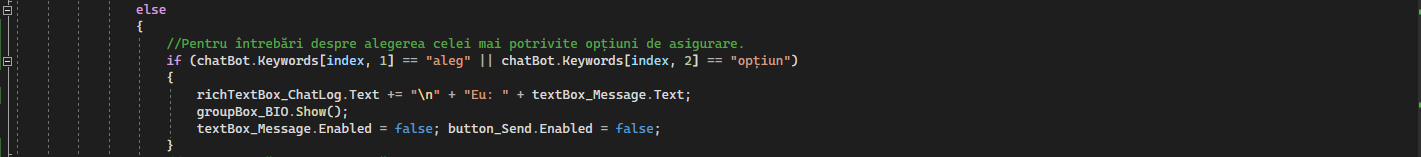
*Anexa 1.1.*

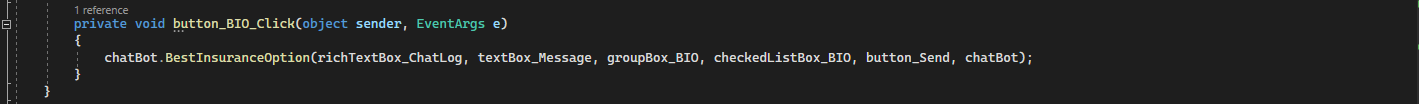


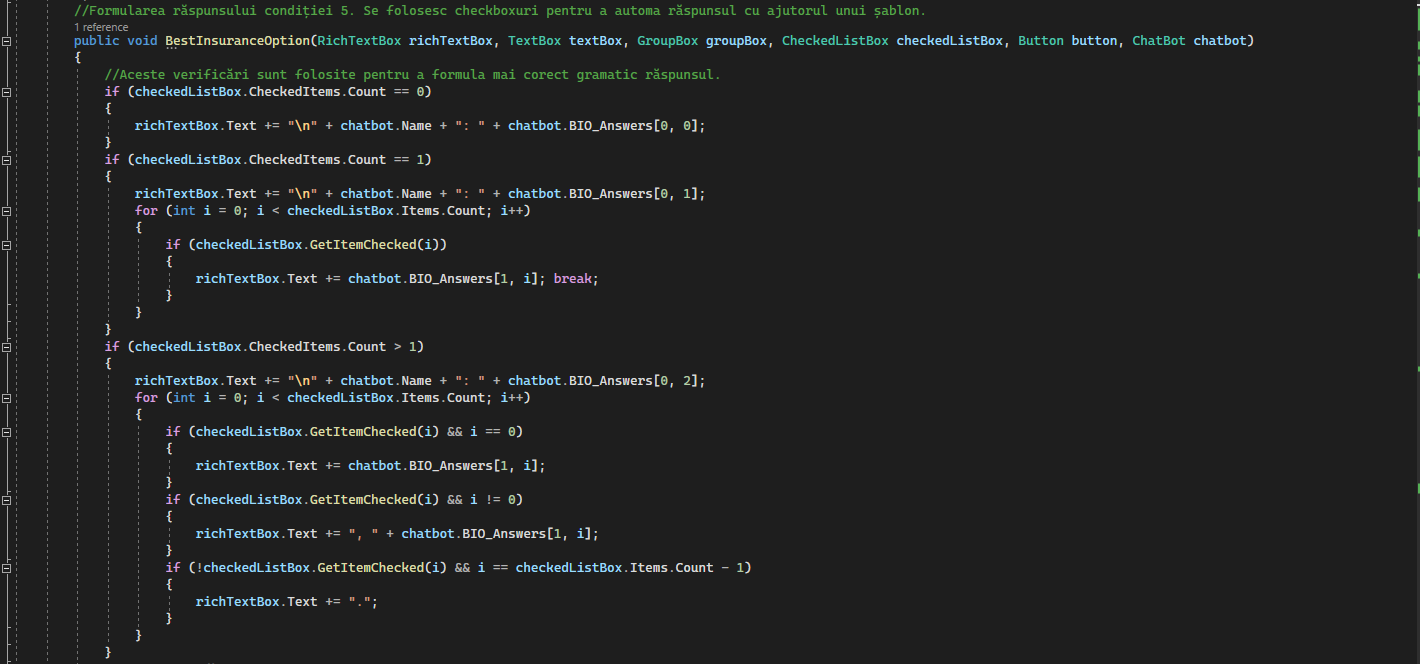
*Anexa 1.2.*

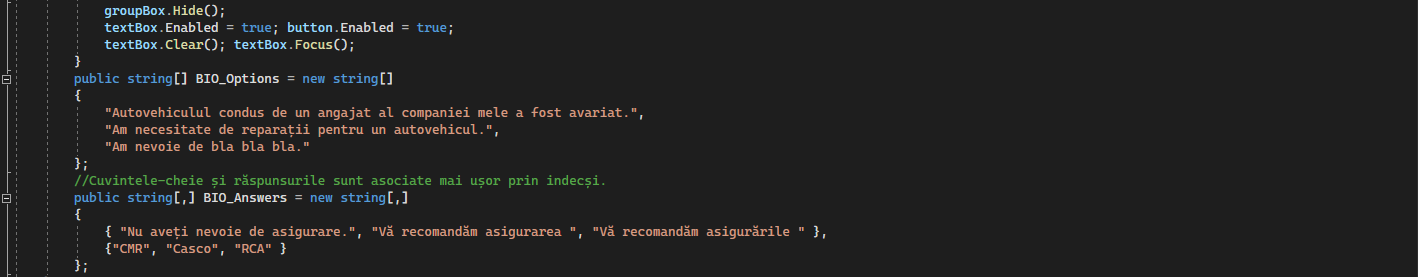


*Anexa 1.3.*









*Anexa 1.4.*

