UNIVERSITATEA “ȘTEFAN CEL MARE”, SUCEAVA

FACULTATEA DE INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR

PROGRAM DE STUDIU CALCULATOARE

”Happy Bee”

Aplicație mobile - manageriere stupina

Student: Profesor coordonator

Andrei Marius-Vasile conf. dr. ing. Schipor Ovidiu-Andrei

Suceava, 2021

Cuprins

[TEMA ȘI MOTIVAȚIA ALEGERII 3](#_Toc75867125)

[I. Capitolul I. Tehnologii utilizate 4](#_Toc75867126)

[I.1. Tehnologii specifice bazelor de date 4](#_Toc75867127)

[I.1.1. Tehnologiile SQL și MySQL 4](#_Toc75867128)

[I.1.2. Elemente specifice bazelor de date 4](#_Toc75867129)

[I.2. Tehnologii specifice mobile 5](#_Toc75867130)

[I.2.1. Software Android 5](#_Toc75867131)

[I.2.2. Java 6](#_Toc75867132)

[I.2.3. XML 6](#_Toc75867133)

[I.3. IDE-uri și platforme utilizate 7](#_Toc75867134)

[I.3.1. FIGMA 7](#_Toc75867135)

[I.3.2. GitLab 8](#_Toc75867136)

[I.3.3. SQL Developer Data Modeler 8](#_Toc75867137)

[I.3.4. Android Studio 8](#_Toc75867138)

[II. Capitolul II. Specificații de implementare (20 pg) 10](#_Toc75867139)

[II.1. Proiectarea aplicației 11](#_Toc75867140)

[II.1.1. Crearea aplicației 11](#_Toc75867141)

[II.1.2. Proiectarea bazei de date (structura bazei de date, constrângeri asupra datelor) 11](#_Toc75867142)

[II.2. Proiectarea codului 11](#_Toc75867143)

[II.2.1. Diagrame UML(clase dezvoltate) 11](#_Toc75867144)

[II.2.2. Diagrame Workflow(logica principala a programului) 11](#_Toc75867145)

[II.2.3. Scenarii de testare 11](#_Toc75867146)

[II.3. Codarea aplicației 11](#_Toc75867147)

[II.3.1. Implementarea bazei de date 11](#_Toc75867148)

[II.3.2. Platformă mobile 12](#_Toc75867149)

[III. Capitolul III. Specificatii de utilizare (20 pagini) 12](#_Toc75867150)

[III.1. Pornirea aplicației 12](#_Toc75867151)

[III.2. Utilizarea aplicației 12](#_Toc75867152)

[IV. Concluzii (1 pagina) 12](#_Toc75867153)

[V. Bibliografie (1 pagina) 12](#_Toc75867154)

[VI. Anexe 13](#_Toc75867155)

# TEMA ȘI MOTIVAȚIA ALEGERII

Tema proiectului pe care am ales-o este denumită “Happy Bee”. Am ales această temă datorită unei curiozități legate de demersul îngrijirii albinelor și pașii pe care un apicultor îi desfășoară din momentul în care se realizează stupina, cât și pașii ulteriori ai acesteia cum ar fi deplasarea albinelor în anumite puncte pentru o producție cât mai bună, până la verificarea și tratarea acestora, dar și evaluarea producției pe baza cântării.

Motivația alegerii temei ar consta în faptul că acest subiect este prea puțin abordat și mulți apicultori sunt doritori să împărtășească sfaturi despre îngrijirea albinelor sau să știe în ce stadiu se află stupina lor în acel moment. Am dorit să vin în ajutorul acestora pentru a le oferi și altă variantă la agenda clasică pe care se ținea evidența stupinei. Această aplicație o să ofere tot ce au nevoie pentru a putea afla mai repede în ce stare se află stupina și tot odată să poată transmite sfaturi și îndrumări celor ce sunt la început de drum și doresc să transmită mai departe această meserie.

În urma studiului de piață a ieșit la iveală o necesitate foarte mare pentru un produs cât mai complet deoarece în acest moment sunt doar câteva aplicații care nu satisfac nevoile apicultorilor.

Multe din acestea sunt incomplete sau nu mai au suport, chiar și greu de utilizat.

Aplicația este alcătuită din două strucuturi diferite: site web educative, blog și “Agendă apicolă” realizată atât pentru dispozitivele mobile cât și pentru platforma web.

# Capitolul I. Tehnologii utilizate

Aplicația mobile de manageriere stupină este alcătuită din toate tehnologiile prezente în figura, Fig.I.1. În continuare vor fi prezentate aceste tehnologii care au stat la baza creării acestui software pentru apicultori .

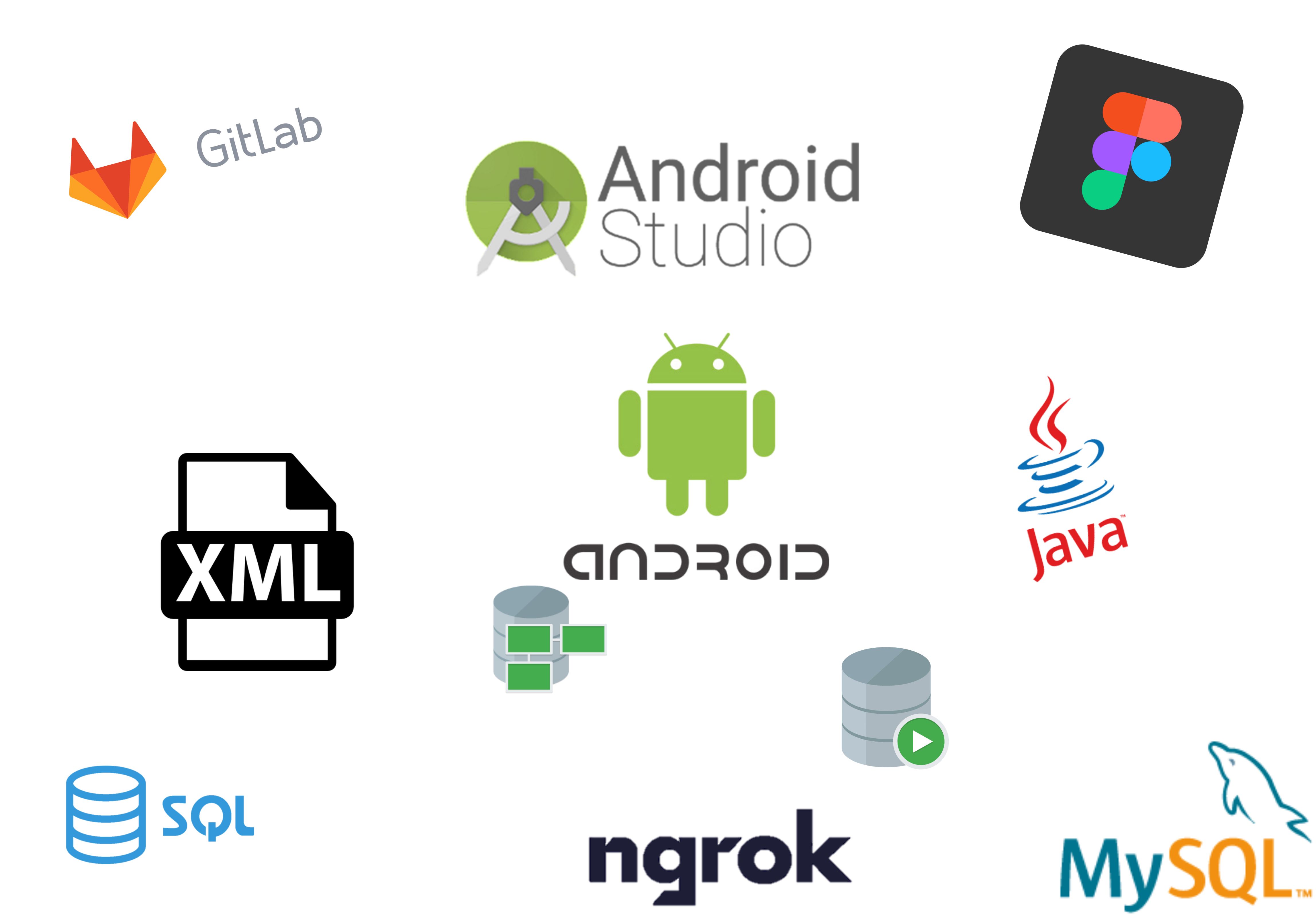


Fig. I.1 Tehnologii și IDE-uri utilizate în cadrul proiectului

## Tehnologii specifice bazelor de date

Deoarece această aplicație avea nevoie de un loc unde să putem stoca datele, am început prin crearea unei baze de date. Aceasta a fost realizate cu ajutorul limbajelor SQL și MySQL. Limbajul SQL a fost utilizat în etapa de conturare a bazei de date, ulterior folosindu-ne de MySQL pentru implementarea si gestiunea acesteia.

### Tehnologiile SQL și MySQL

**SQL** este un limbaj de programare care permite relaționarea intre un sistem și baza de date. Acesta este un limbaj neprocedural, bazat pe algebra relaționară. Cu ajutorul acestui limbaj se pot realiza lucrări de introducere, ștergere și actualizare a datelor. Acest limbaj este răspândit la scală largă datorită beneficilor pe care le oferă**.**

**MySQL** este un model de administrare a bazelor de date relaționare. MySQL folosește un dispozitiv care are la baza sa un sistem precum Windows sau Linux și cu ajutorul acestuia se poate pune în funcțiune o bază de date relaționară care să poată gestiona anumite date cu care aplicația lucrează.

### Elemente specifice bazelor de date

Elementele specifice bazelor de date sunt:

* **Sistemul de gestiune a bazelor de date (SGBD)** este o bază de date care poate fi adusă la zi prin instrucțiuni de tip UPDATE, modificată cu instrucțiunea ALTER sau pentru vizualizarea de rapoarte.

Tabelele din componența unui SGBD pot avea următoarele legături:

* **1:1 –** pentru fiecare înregistrare din tabelul 1 i se potrivește doar o singură înregistrare din tabelul 2**;**
* **1:N –** pentru fiecare înregistrare din tabelul 1 i se potrivește una sau mai multe înregistrări din tabelul 2, dar pentru fiecare înregistrare din tabelul 2 îi corespunde o singură înregistrare din tabelul 1**;**
* **M:N –** pentru fiecare înregistrare din tabelul 1 i se potrivește una sau mai multe înregistrări din tabelul 2 și invers**;**
* Pașii de elaborare a unui **SGBD** sunt:

1. Examinarea mediului în care va fi utilizat SGBD-ul;

2. Schițarea diagramei ERD;

3. Programarea logică a bazei de date;

4. Programarea fizică a bazei de date;

5. Realizarea bazei de date.

* Diagrama ERD este utilizată pentru a pune în evidență legăturile ce se stabilesc între tabele și a câmpurilor aflate în acestea.

## Tehnologii specifice mobile

### Software Android

Android este un software care este disponibil pe majoritatea dispozitivelor mobile cu touchscreen. Aceasta are la bază un kernel Linux fiind dezvoltat de “Open Handset Alliance”. Primul telefon care avea la bază Android a fost HTC Dream în anul 2008. Acesta urmând a fi startul unei noi ere a telefoanelor inteligente.

Variante ale sistemului Android au fost:

* 1.5 nume de cod Cupcake a fost primul Android intalat pentru prima data pe HTC dream. De aici începând cu adevărat acel salt major în industria telefoanelor;
* 1.6 – Donut prima versiune care ofera suport pentru diferite dimensiuni ale ecranelor dispozitivelor;
* 2.1 – Éclair oferea o bara de căutare, funcții pentru cameră și tool-uri pentru a customiza pozele;
* 2.2 – Froyo oferea îmbunătățiri asuprea vitezei sistemului de operare, tethering USB pentru depanarea de aplicații, hotspot și multe altele;
* 2.3 – Gingerbread aducea un design nou, oferea suport pentru voice over internet protocol,NFC și afișarea GIF-urilor;
* 3.0 – Honeycomb a fost utilizată în special pe tablete. Această versiune a fost mai mult o concentrarea asupra ceea ce înseamnă experiență a utilizatorului;
* 4.0 – IceCreamSandwich introduce posibilitatea de realiza captura de ecran, fereastra noua de widgeturi și Android Beam care permitea dispozitivelor să comunice cât mai repede când acestea se află unul lângă celălalt;
* 4.1 – JellyBean aduce sprijin pentru TLS, dezactivarea notificărilor pentru aplicații și multe altele;
* 4.4 – KitKat îmbunătățește interfața și securitate, schimbare butoanelor, introducerea senzorilor de mișcare și multe altele;
* 5.0 – Lollipop aduce schimbări majore odată cu introducerea suportului pentru procesoare de 64 bit, introducerea Proiectului Volta care realiza creșterea autonomiei bateriei, ecranul de blocare prezintă comenzi rapide;
* 6.0 – Marshmallow a adăugat suportul pentru USB-C, back-up automat in contul Google Drive, funcția care introduce aplicațiile în modul așteptare;
* 7.0 – Nougat introduce folosirea a două aplicații simultan, modul de reducere al nuanței de albastru pentru odihnirea ochiilor, posibilitatea de modificare a rezoluției;
* 8.0 – Oreo aduce suport PIP, suport audio AAC, o autonomie mai mare;
* 9.0 – Pie introduce noi tranziții, ecran de blocare cu Always-On, performanțe ridicate NFC, posibilitatea de a răspunde la mesaje din bara de notificări;
* 10 – Android 10 aduce navigarea prin gesturi mult îmbunătățită, mod întunecat, suport pentru formatul video HDR10+;
* 11 – Android 11 permite înregistrarea ecranului, Android Auto prin conexiune Wi-Fi, afișarea istoricului notificărilor;
* 12 – Android 12 oferă posibilitatea de partajare a Wi-Fi mult mai ușor, suport AVIF, îmbunătățirea gesturilor.

### Java

Java este un limbaj de programare care este bazat pe lucrul cu clase, orientat-obiect, care este cunoscut pentru faptul că este un limbaj care datorită simplității acestui se scrie cod o singură dată și poate fi rulat pe mai multe platforme care suportă Java.

Java a evoluat ințial cu ajutorul lui James Gosling în compania Sun Microsystems, fiind lansat în 1995. Implemetarea de referință este OpenJDK JVM, care este un software gratuit, JVM este implicit folosit pe sistemele Linux.

Ultima versiune fiind Java 16, care are la bază Java 11 care oferă suport îndelungat. Ultima versiune care oferă suport îndelungat este Java 8 care are cost zero pentru utilizare personală. Java 8 și Java 11 oferă suport și în ziua de azi pentru actualizări de securitate.

Rularea programelor Java pe orice dispozitiv este realizată cu ajutorul unui fișier numit Java bytecode. Acest fișier compilat poate rula pe orice calculator care are instalat Java Runtime Environment (JRE) sau prin intermediul browserului.

Compilarea programelor în Java este realizată pe principiul Just-in-Time(JIT), compilând coduri de octeți în cod mașină. Datorită acestei compilări a crescut viteza de compilare și scăderea memoriei folosite comparativ cu programele C++.

Cele patru programe Java furnizate de Oracle sunt:

* Java Card – carduri cu cip;
* Java Platform, Micro Edition(Java ME) – PDA, telefoane mobile;
* Java Platform, Standard Edition(Java SE) – PC;
* Java Platform, Enterprise Edition(Java EE) – Servere.

Câteva medii de dezvoltare care utilizează Java sunt:

* Eclipse;
* NetBeans;
* IntelliJ IDEA;
* BlueJ;
* DrJava;
* JCreator.

### XML

XML (Extensible Markup Language) este un limbaj de marcare care utilizează niște reguli pentru a codifica unele fișiere într-un mod care să poată fi înțeles de către om dar și de către mașină.

XML este o versiune mai mult mai ușoară a limbajului SGML și are rolul de emite date între aplicații aflate pe internet.

Acest limbaj de marcare folosește tag-uri pentru a delimita unele aspecte legate de mesajul trimis sau de interfețele create. Aceste tag-uri sunt definite de către un autor, ulterior fiind interpretate de programul aflat pe aceea mașină.

Principalele avantaje ale XML-ului sunt:

* Se pot construi oricând noi taguri pentru a defini o nouă acțiune:
* Se verifică corectitudinea datelor care au fost transmise prin acest limbaj definit de utilizator;
* Oamenii care folosesc acest limbaj pot redefini oricând tagurile pentru ceea ce au nevoie;
* Acest limbaj este simplu și ușor;
* Se poate edita textul fără avea unu editor text specializat.

Versiuni ale XML:

* XML 1.0 a apărut în anul 1998. Acesta este folosit și în ziua de azi;
* XML 1.1 a apărut în anul 2004. Modificările care le aduce această versiune permite utilizare caracterelor de finalizare a liniei pentru platformele EBCDIC și care folosesc caractere pentru Unicode 3.2. Această versiune nu este folosită pe scară largă și este recomandată a fi folosită în cazul în care este nevoie de acele caractere specifice.

## IDE-uri și platforme utilizate

Proiectul a fost realizat cu ajutorul platformelor:

* **Figma** în care s-a realizat designul aplicației și a fost utilizat pentru lucrul în echipă datorită posibilității de a muncii în timp real;
* **GitLab**  a fost utilizat pentru salvarea proiectului online și pentru posibilitatea de a salva proiectul pe versiuni;
* **SQL Developer Data Modeler** a fost utilizat pentru crearea codului SQL pentru baza de date utilizată în interiorul aplicației Android;
* **Android Studio** a fost utilizat pentru a putea crea aplicația mobile.

### FIGMA

Figma este un utilitar care permite crearea de prototipuri și schițe prin intermediul browserului web. Acesta este disponibil pe dispozitivele cu Windows și MacOS. Figma a aparut în anul 2016, mai precis în luna septembrie, fiind realizat de către doi studenți Dylan Field și Evan Wallace de la Universitatea Brown.

Aplicația Figma oferă unelte pentru design în funcție de proiectul pe care l-ai stabilit(mobile sau web). Fereastra principală are trei secțiuni:

* Design – această secțiune oferă posibilități de aliniere a imaginilor, deplasarea obiectelor stânga dreapta, rotunjirea colțurilor aplicarea de constrângeri, culori, efecte, moduri de export și multe altele;
* Prototype – aici se poate selecta device-ul din lista oferită pentru a putea avea o vedere asupra aspectului pe un anumit dispozitiv, putând fiind schimbată orientarea afișării(vertical sau orizontal), background pe care sa fie afișată simularea, numărul de cadre pe secundă;
* Inspect – acestă secțiune oferă moduri de generare a codurilor pentru ferestrele create cum ar fi Android, iOS, CSS.
* Share – permite distribuirea planșei de lucru;
* Această aplicație mai permite și opțiune precum: selectare, mutare, text, schițare, dar și mod de comentarii pentru a adăuga detalii sau idei de către tine sau ceilalți membri ai echipei.

### GitLab

GitLab este o aplicație care are la baza sa legături cu web-ul pentru dezvoltori de aplicații care pun la dispoziție un management de depozitare a proiectelor. Acestă aplicație face posibilă salvarea mai ușoară a versiunilor. Această funcționalitate permite găsirea bug-urilor mai rapidă, memorarea diferitelor versiuni și selectarea versiunilor.

Câteva comenzii de GitLab:

* git clone – clonează proiectul curent;
* git status – afișează modificările care au fost realizate;
* git commit –m – crează un mesaj nou de la salvarea trecută pe git;
* git add – adaugă fișierul la salvarea trecută;
* git push origin – salvarea proiectului pe versiunea dorită.

Acest software a fost dezvoltat de Dmitriy Zaporozhets și Valery Sizov care erau de origine ucraineană.

GitLab mai oferă posibilitatea de crearea de task-uri pentru membri echipei. Dimensiunea fișierelor este de maxim zece GB și nu oferă restricți asupra datelor, un singur fișier poate avea zece GB.

### SQL Developer Data Modeler

SQL Developer Data Modeler este un software care permitea realizarea de SGBD. Aceast produs este oferit de Oracle Corporation în mod gratuit. Acesta mai oferă posibilitatea de gestionare a bazei de date, executarea interogărilor și a scripturilor, realizarea de rapoarte, migrare a datelor, dar și modelarea datelor.

Ultima versiune stabilă a acestui software este 21.1, fiind disponibilă pentru macOS, Linux și Windows.

Avantaje ale SQL Developer Data Modeler:

* Oferă asistență pe domeniu;
* Permite utilizarea modelelor logice, relaționare și fizice oferind tot odată și reguli de proiectare pentru acestea;
* Poate fi folosit și offline;
* Oferă un depozit de rapoarte pentru interogări;

Acest produs este folosit de ingineri, arhitecți de întreprinderi, dezvoltatori de aplicații și proiectanți pentru o a realiza o baza de date cât mai bine structurată și datorită multitudinilor de unelte care fac munca mult mai ușoară.

### Android Studio

Android Studio este un mediu de dezvoltarea pentru aplicații Android. Prima versiune stabilă a fost lansată în anul 2014. Android Studio avea la baza sa software-ul IntelliJ IDEA care era realizat de JetBrains. Platfomele care suportă acest software de dezvoltarea a aplicațiilor pentru Android sunt Windows, MacOS sau Linux.

Limbajele care sunt suportate în acest moment în Android Studio sunt: Java și Kotlin. Pentru ca acesta să poată rula în condiții optime este necesar opt GB de RAM, opt GB de spațiu liber pe disk, Java Development Kit 8 și o rezoluție minimă de 1280x800.

Pașii pentru dezvoltarea aplicației:

* Realizarea setărilor pentru dezvoltarea aplicației cum ar fi tipul de template folosit, versiunea de android minimă, configurarea emulatorului sau pornirea modului de dezvoltator pentru rularea aplicației pe dispozitivul Android;
* Configurarea și dezvoltarea proiectului face referire la uneltele folosite de aplicație și fișiere de cod sursă;
* Testarea, rularea și conceperea aplicației – Acestă etapă face referire la crearea fișierului care poate fi rulat pe emulator sau pe dispozitivul cu Android;
* Lansarea aplicației – aici se realizează setările pentru a putea lansa aplicația publicului.

Android Studio organizeză fișierele în module formând în final un fișier apk. Structura unui fișier apk este următoarea:

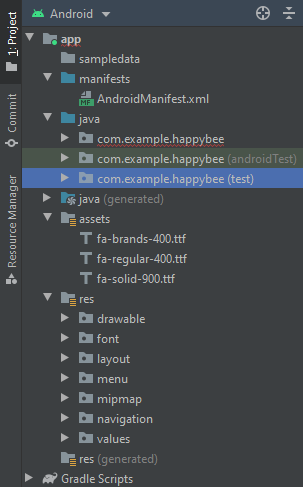


Figura I.3.4.1 Structura unui proiect în Android Studio

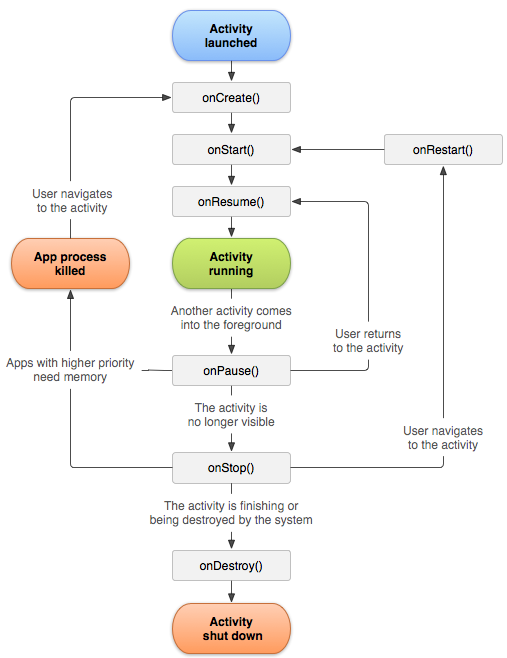
* Android Manifest.xml – este fișierul în care se află o declarație a tuturor permisiunilor și a componentelor aflate în întreaga aplicație;
* Java – în această secțiune se află codurile java sub formă de clase și în care se regăsește logica aplicației pentru fiecare obiect;
* Java(Generated) – se află un fișier de configurări în care este precizat versiunea acesteia, id-ul și modul în care rulează aplicația;
* Assets – fișier ce conține fonturi ce au fost utilizate în aplicație;
* Res – este directorul de resurse care legate de afișaj, font-uri și multe altele. Acest fișier are următoarele subansamble:
* Drawable – în acest fișier se află tot ceea ce ține de alcătuirea design-ului cum ar fi: iconițe, logo, codul pentru rotunjirea colțurilor butoanelor și a campurilor cu paleta de culori aferentă;
* Font – aici se află fonturile folosite pentru textul din aplicație;
* Layout – în această secțiune a fost create tot ce ține de aspect, mai exact ferestrele care vor fi afișate în aplicație;
* Menu – aici a fost creat tot ce ține de meniul prezent în aplicație;
* Mipmap – locul unde se află toate iconițele necesare pentru lansarea aplicației în Google Play;
* Navigation – ferastră necesară pentru navigarea în meniu;
* Values – declararea de culori, stiluri și iteme folosite în diferite câmpuri aflate în ferestre;
* Gradle Scripts – declararea bibliotecilor folosite în aplicație, dar și dependențele și versiunea minima de Android suportată.

Elemente specifice oricărei aplicații Android sunt:

* Activitățile sunt niște componente fundamentale pentru orice aplicație Android. Acestea sunt defapt niște ferestre care fac legătura cu utilizatorul. Aceste ferestre aduc informația pe care utilizatorul o dorește în acel moment. Cele mai multe aplicații conțin mai multe ferestre de activități și ele se apelează în funcție de nevoia utilizatorului.

O activitate are la baza sa 4 stări:

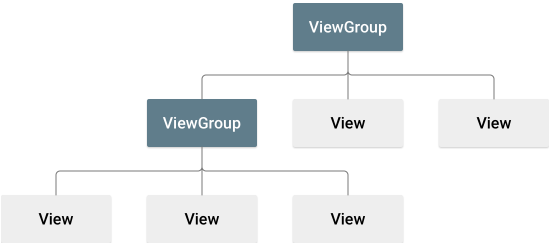
1. Running – atunci când aplicația este pe ecran și utilizatorul poate realiza acțiuni în acel moment;
2. Visible – aplicația se află întru-un stadiu de rulare da nu mai este în prim plan. În acel moment fiind altă aplicație care rulează pe ecranul principal;
3. Stopped sau Hidden – este starea aplicației atunci când ea mai stochează date dar ea nu se mai află deloc în câmpul vizual al utilizatorului. În această stare aplicația va fi oprită de sistem în cazul în care aceasta va cere utilizarea excesivă de memorie sau un alt proces are nevoie de mai multe resurse;
4. Destroyed – în cazul în care aplicația a fost într-o stare de distrugere aceasta va trebui să repornească complet la redeschiderea ei. Cauzele pentru care o aplicație se poate introduce în acest stadiu este acela că folosește foarte multe resurse ale dispozitivului sau încetinește alt proces esențial în rularea optimă a dispozitivului.



Figură I.3.4.2 Ciclul de viață a unei aplicații Android

Metodele care se utilizează în ciclul de viață a unei aplicații Android sunt:

* onCreate( ) – este utilizat atunci când activitatea nu este creată și trebuie să apară pe ecran. Neapărat după metoda onCreate() tebuie să fie utilizat onStart( );
* onRestart( ) – este utilizat dacă aplicația este oprită. După onRestart( ) trebuie utilizat onStart( );
* onStart( ) – folosit pentru a aduce activitatea în prim plan. Utilizat neapărat cu onResume( ) și onStop( );
* onResume( ) – această activitate se reia când utilizatorul folosește din nou aplicația și este urmat de onPause( );
* onPause( ) – este utilizat pentru a scoate aplicația din prim plan. Pentru a o readuce în prim plan se folosește metoda onResume( ) sau pentru oprirea ei se utilizează onStop( );
* onStop( ) – folosit în momentul în care nu se mai vrea aplicația a fi activă. Pentru a readuce la viață aplicația se utilizează onRestart( ) sau pentru a o opri complet se folosește onDestroy( );
* onDestroy( ) – utilizat atunci când aplicația se dorește a fi oprită complet. Această comandă are un efect reversibil. Odată realizată aplicația este oprită complet. Metodă utilizată preponderent în vederea economisirii de spațiu.
* Layout-urile sunt reprezentate sub formă dreptunghiulară și se utilizează pentru a caracteriza structura unei interfețe.



Figură I.3.4.3 Structura unui ViewGroup în Android Studio

Obiectul View se ocupă cu afișarea pe ecran a evenimentelor. ViewGroup face parte din clasa View, fiind utilizat pentru crearea de noi View-uri sau alte ViewGroup-uri.

Layout-urile scrise în limbajul XML. Aici se pot unii sau separa diferite secțiuni ale ecranului.

Tipuri de layout-uri utilizate în Android Studio sunt:

* Linear Layout – toate ansamblele sunt vizualizate într-un singur mod, cum ar fi: drept sau longitudinal;
* Relative Layout – toate ansamblele sunt private în poziții relative;
* Table Layout – câmpurile se vizualizează pe rânduri și coloane;
* Absolute Layout – se utilizează poziția câmpurilor deja situate în acea ferastră;
* Frame Layout – utilizat pentru afișarea unei alte ferestre pe acel ecran;
* List View – poziția elementelor pe ecran fiind sub formă de listă;
* Grid View – situarea elementelor sub formă de tabel bidimensional.

Atribute specifice pentru obiectele de tip View:

* Id – se atribuie un id unic deoarece nu au voie să se suprapună aceste id-uri;
* Layout\_witdh – stabilește lățimea;
* Layout\_height – stabilește înălțimea;
* Layout\_marginTop – stabilește un spațiu deasupra view-ului;
* Layout\_marginBottom – stabilește un spațiu dedesubtul view-ului;
* Layout\_marginLeft – stabilește un spațiu în stânga view-ului;
* Layout\_marginRight – stabilește un spațiu în dreapta view-ului;
* Layout\_constraintEnd\_toEndOf – stabilește constrângera față de obiectul terminal;
* Layout\_constraintHorizontal\_bias – stabilește constrângera față de marginea view-ului;
* Layout\_constraintStart\_toStartOf – stabilește constrângera față de obiectul de start;
* Layout\_constraintTop\_toTopOf – stabilește constrângerea față de marginea de sus a view-ului etc.



Figură I.3.4.4 Atribute specifice pentru obiectele de tip View

* Android Manifest este locul în care se face legătura aplicației cu software-ul Android, chiar și cu utilizatorul. Aici se pot stabili permisiunile, pachetul din care face parte, versiunea minimă a SDK-ului, modelul de hardware folosit, biblioteci pentru diferite utilități.

Permisiunile satisfac nevoile utilizatorilor pentru a putea avea acces la anumite obiecte alea sistemului. Odată ce acestea sunt restricționate, posibilitatea de a le activa este foarte mică.



Figură I.3.4.5 Permisiuni Android

Apartența la pachet este necesară deoarece aplicația trebuie să știe în orice moment cu ce fișiere trebuie să lucreze.

Precizarea versiunii minime este necesară pentru a ști cu ce aplicații și obiecte poate lucra acea aplicație. Datorită evoluții continue a versiunilor de Android, multe din metode sau funcții nu mai sunt compatibile.

Specificarea hardware-ului folosit fiind și ea necesară deoarece un software foarte nou va rula cu foarte multe probleme pe un hardware învechit. Astfel elimându-se problemele care pot apărea din cauza unui hardware prea slab.

Adăugarea bibliotecilor pentru anumite obiecte folosite în aplicație ne poate scuti de timpul pierdut încercând să realizăm un obiect ce a fost realizat deja.

* Fire de execuție

O aplicație Android va avea mereu un proces care va rula. În momentul în care aplicația pornește acest proces ea va rula doar pe un singur fir de execuție. În cazul în care acea aplicație pornește altă acțiune va folosi același proces și va utiliza și același fir de execuție.

Multitaskingul este alcătuit din aceste operatiuni de procesare și fire de execuție. Procesele și firele de execuție sunt gestionate de sistemul de operare și acesta decide când să fie oprite sau să ruleze.

# Capitolul II. Specificații de realizare a proiectului

“Happy Bee” Aplicație mobile - manageriere stupina a fost implementată trecând prin următoarele etape:

1. Stabilirea unui țel și aranjarea proiectului;
2. Selecția tehnologiilor;
3. Schițarea programului;
4. Realizarea și testarea software-ului.

Etapele principale au fost înfățișați în primele capitole, etapele finale fiind prezentate în cele ce urmează.

## Schițarea aplicației

Schițarea aplicației este necesară pentru a putea vizualiza toate aspectele ce se doresc atinse pentru ca proiectul să fie de un impact imens în starea finală, dar și pentru funcționarea cât mai optimă și lipsită de erori. În continuare se vor prezenta pașii ce au fost realizați pentru crearea aplicației și stadiul în care se află aplicația.

### Crearea aplicației

Figma este utilitarul cu care s-a realizat design-ul aplicației mobile. Această unealtă a pus la dispoziția utilizatorului setări cum ar fi: mărimea ferestrei aplicației prin selectarea tipului de dispozitiv pus la dispoziție, selectarea de fonturi, culori și iconițe pentru meniurile și butoanele aplicației, constrângeri asupra elementelor aplicației, dar și pozibilitatea de simularea a apariției schițelor pe ecranul unui dispozitiv simulat în aceasta.

Iconițele utilizate în aplicație au fost preluate de pe Font Awesome pentru a putea aduce designul cât mai aproapre de ceea ce ne doream.



Fig. II.1 Iconițe utilizate în Figma



Fig. II.2 Design aplicație mobile

### Țelul aplicației mobile

După realizarea designului aplicației a început urmărirea țelului acesteia. În urma unei analize destul de amănunțite a ieșit la iveală faptul este nevoie de un mediu în care să salvă datele pe care apicultorii le vor furniza prin intermediul aplicației, dar au apărut și următoarele aspecte legate de modul în care aceasta trebuie să funcționeze. Elementele sunt următoarele:

* Fiecare opțiune din meniu va deschide o fereastră în care va fi afișată stupii din stupini cu caracteristicile pentru fiecare opțiune în parte;
* Utilizatorul va trebui să primească notificări pe baza datelor introduse în aplicație;
* Secțiunea de feedback trebuie să fie prezentă pentru a anunța dezvoltatorul pentru diferitele erori ale programului;
* Menținerea datelor în memoria cache a sistemului pentru a putea salva datele ulterior la prezența unei conexiuni la internet;
* Preluarea locației folosind hărți.

După realizarea elementelor necesare aplicației am dedus că am avea nevoie de o bază de date pentru a salva aceste date pentru ca aplicația să poată rula și oferi cele mai bune performanțe, dar și salvarea acestora într-un mediu care să aibă o structură bine definită.

### Proiectarea bazei de date

În această etapă de proiectarea a bazei de date am pornit prin stabili ce nevoie au apicultori

## Proiectarea codului

### Diagrame UML(clase dezvoltate)

### Diagrame Workflow(logica principala a programului)

### Scenarii de testare

## Codarea aplicației

### Implementarea bazei de date

### Platformă mobile

# Capitolul III. Specificatii de utilizare (20 pagini)

## Pornirea aplicației

## Utilizarea aplicației

# Concluzii (1 pagina)

# Bibliografie (1 pagina)

<https://www.figma.com/>

<http://www.eed.usv.ro/~mdanub/cursuri/>

<https://www.oracle.com/ro/database/what-is-database/>

<https://ro.m.wikipedia.org/wiki/MySQL>

<https://en.wikipedia.org/wiki/Figma_(software)>

<https://ro.wikipedia.org/wiki/XML>

<https://developer.android.com/guide/topics/ui/declaring-layout>

<https://www.tutorialspoint.com/android/android_user_interface_layouts.htm>

<https://developer.android.com/guide/components/processes-and-threads>

# Anexe