Cassandra’s Tome of Secrets

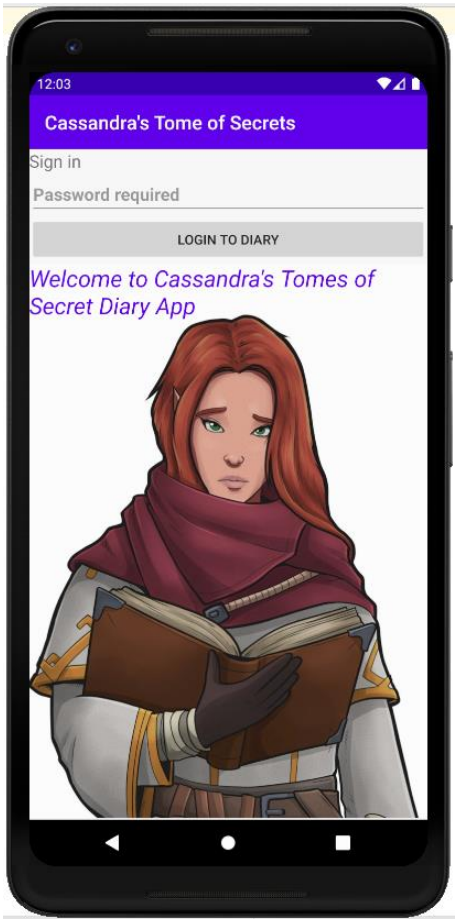
Ez a mobil alkalmazás egy napló alkalmazás, amely a biztonságos naplóírást teszi lehetővé. Az alapvető funkciók mellett a naplóbejegyzések jelszóval vannak védve. Az alapértelmezett jelszó az „admin”, azonban ezt a jelszót a későbbiekben testre szabhatja a felhasználó az appon belül.

Kezdő képernyő

A bejelentkező felületen egy animáció hajtódik végre indításkor, amikor lassan megjelenik az App nevét adó Cassandra figura és üdvözli a felhasználót.

```
private boolean validatePassword() {
    String passwordIn = inputPassword.getText().toString();
    if (passwordIn.equals(sharedPreferences.getString( key: "password", defValue: null)))
    {
        return true;
    }
    return false;
}
public void HandleAnimation(View view){
    ObjectAnimator animator = ObjectAnimator.ofFloat(view,View.ALPHA, ...values: 0.0f,1.0f);
    animator.setDuration(animationDuration);
    AnimatorSet animatorSet = new AnimatorSet();
    animatorSet.playTogether(animator);
    animatorSet.start();
}
```

```
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    sharedPreferences = getApplicationContext().getSharedPreferences( name: "Preferences", mode: 0);
    editor=sharedPreferences.edit();
    if(!sharedPreferences.contains("password")){
        editor.putString("password","admin");
        editor.commit();
    }
    inputPassword = (EditText)findViewById(R.id.password);
    loginView= (ImageView)findViewById(R.id.loginImage);
    loginView.setImageResource(R.drawable.tome_of_secrets);
    greeting=(TextView)findViewById(R.id.greeting);
    HandleAnimation(loginView);
    HandleAnimation(greeting);
    configureLoginButton();
}
```



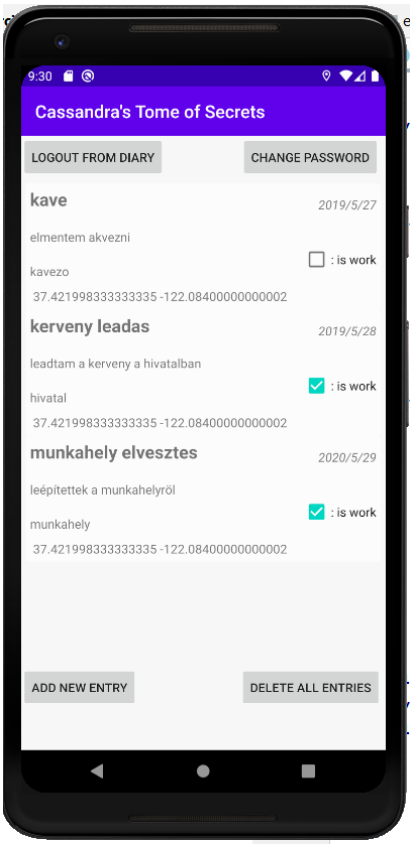
Napló bejegyzések listája képernyő

Sikeres jelszómegadás után az applikáció a naplóbejegyzések listáját tartalmazó Activity-re (ListEntriesActivity) kerül, ahol felül és alul két két funkció gomb található, illetve középen egy RecycleView-ben az adatokat listába rendezve jeleníti meg az applikáció. A RecycleView-ban az elemek CardView-ban vannak tárolva, és a bejegyzések címét, leírását, helyszínét, időpontját, azt hogy munka-e vagy sem illetve a koordinátákat tartalmazzák (A címen és a leíráson kívül a többi

opcionális). A négy gomb a kijelentkeztető gomb, ez esetben újra be kell majd lépni az applikációba, a jelszó megváltoztatása gomb, ez esetben meg lehet változtatni azt a jelszót, ami a belépéshez szükséges, az új naplóbejegyzés létrehozásával egy Activity nyílik meg ahol új bejegyzést adhatunk hozzá a meglévőkhöz, az összes törlése gomb pedig törli az összes naplóbejegyzést. Törölni egyesével is lehet a naplóbejegyzéseket, ehhez jobbra vagy balra kell tolni az adott bejegyzést, ezáltal törölve az adatbázisból.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_list_entries);
    listOfEntriesRecycle=(RecyclerView)findViewById(R.id.listOfEntriesRecycle);
    dataBase = Room.databaseBuilder(getApplicationContext(),AppDataBase.class, name: "diaryDatabase").fallbackToDestructiveMigration().allowMainThreadQueries().build();
    entries = dataBase.entryDao().getAllEntry();
    listOfEntriesLayoutManager=new LinearLayoutManager( context: this);
    listOfEntriesAdapter=new EntryAdapter(entries);
    listOfEntriesRecycle.setLayoutManager(listOfEntriesLayoutManager);
    listOfEntriesRecycle.setAdapter(listOfEntriesAdapter);
    ItemTouchHelper swipeToDelete = new ItemTouchHelper(new ItemTouchHelper.SimpleCallback( dragDirs: 0, swipeDirs: ItemTouchHelper.LEFT|ItemTouchHelper.RIGHT ) {
        @Override
        public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
            return false;
        }

        @Override
        public void onSwiped(@NonNull RecyclerView.ViewHolder target, int direction) {
            int position = target.getAdapterPosition();
            Entry entry = entries.get(position);
            dataBase.entryDao().delete(entry);
            entries.remove(position);
             listOfEntriesAdapter.notifyDataSetChanged();
        }
    });
    swipeToDelete.attachToRecyclerView(listOfEntriesRecycle) ;
    configureAddNewButton();
    configureDeleteAllButton();
    configureLogoutButton();
    configureChangePasswordButton();
}
```



Új naplóbejegyzés létrehozása

Az Add new Entry-re kattintva egy újabb Activity nyílik meg, ahol megadhatjuk az új bejegyzés címét, tartalmát, a cselekmény helyszínét, időpontját és még akár a jelenlegi koordinátákat is hozzá csatolhatjuk. Ezután visszakerülünk a lista nézetre és a megadott adatokkal kitöltött teret látunk. Fontos, hogy a cím és a leírás kötelező, így annak hiányában a bejegyzés mentésekor hibát jelez nekünk a rendszer.

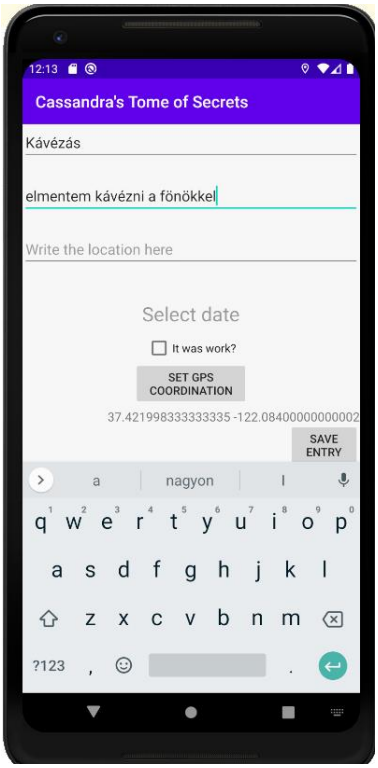
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_new_entry);
    writeNewEntry = (EditText) findViewById(R.id.writeEntry);
    writeNewEntryTitle = (EditText) findViewById(R.id.newEntryTitle);
    writeNewEntryPlace = (EditText) findViewById(R.id.placeEntry);
    isNewEntryWork = (CheckBox) findViewById(R.id.isWorkCheck);
    displayDate = (TextView) findViewById(R.id.newEntryDate);
    displayDate.setOnClickListener((v) → {
            Calendar calendar = Calendar.getInstance();
            int year = calendar.get(Calendar.YEAR);
            int month = calendar.get(Calendar.MONTH);
            int day = calendar.get(Calendar.DAY_OF_MONTH);
            DatePickerDialog datePickerDialog = new DatePickerDialog( context: AddNewEntryActivity.this, android.R.style.Theme_Holo_Dialog_MinWidth,
                    onDateSetListener, year, month, day);
            datePickerDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
            datePickerDialog.show();
    });
    configureLocation();
    dataBase = Room.databaseBuilder(getApplicationContext(), AppDataBase.class,  name: "diaryDatabase").fallbackToDestructiveMigration().allowMainThreadQueries().build();
    onDateSetListener = (OnDateSetListener) (view, year, month, dayOfMonth) → {
            month = month + 1;
            Log.d( tag: "AddNewEntryActivity",  msg: "onDateSet: " + year + "/" + month + "/" + dayOfMonth);
            String date = year + "/" + month + "/" + dayOfMonth;
            displayDate.setText(date);
    };
    configureSaveButton();

}
```

```
private void configureSaveButton(){
    Button saveButton = (Button)findViewById(R.id.SaveNewEntryButton);
    saveButton.setOnClickListener((v) → {
            if(TextUtils.isEmpty(writeNewEntryTitle.getText())){
                writeNewEntryTitle.setError("Title is required to make new entry!");
                return;
            }
            if(TextUtils.isEmpty(writeNewEntry.getText())){
                writeNewEntry.setError("Journal entry is meaningless without the description!");
                return;
            }
            Entry newEntry = new Entry(writeNewEntryTitle.getText().toString(),
                    writeNewEntry.getText().toString(),writeNewEntryPlace.getText().toString(),
                    displayDate.getText().toString(),isNewEntryWork.isChecked(),coordinates.getText().toString());
            int id = ListEntriesActivity.entries.size();
            newEntry.setId(id);
            dataBase.entryDao().insertAll(newEntry);
            ListEntriesActivity.addToEntries(newEntry);
            finish();
    });
}
```

Házi feladat
Név: Csik Levente

Mobil és Webes szoftverek
Neptun: VR6H0R



Perzisztens adattárolás

Az adatok tárolásához kellett egy Entry-osztályt, egy DAO-t és egy Adatbázist létrehozni, amelyek a Room segítségét felhasználva elmentették az adatainkat. Első lépésként az Entry osztályt @Entity jelzővel láttam el, majd az id-ját @PrimaryKey-jel, a többi adatját pedig egy-egy @ColumnInfo-val
Az EntryDao interfészben a kiadható parancsokat, getAllEntry, deleteAll, delete és insert parancsokat definiáltam.
Az adatbázis osztályban pedig létrehoztam egy listát az Entry-knek

```
@Dao
public interface EntryDao {
    @Query("SELECT * FROM entry")
    List<Entry> getAllEntry();

     @Insert
    void insertAll(Entry... entries);

     @Query("DELETE FROM entry")
    void deleteAll();

    @Delete
    void delete(Entry entry);
}
```

```
@Database(entities = {Entry.class},version = 2)
public abstract class AppDataBase extends RoomDatabase {
    public abstract EntryDao entryDao();
}
```

Azután az adatbázist inicializálni kellett a listán belül

```
listOfEntriesRecycle=(RecyclerView)findViewById(R.id.listOfEntriesRecycle);
dataBase = Room.databaseBuilder(getApplicationContext(),AppDataBase.class, name: "diaryDatabase").fallbackToDestructiveMigration().allowMainThreadQueries().build();
entries = dataBase.entryDao().getAllEntry();
listOfEntriesLayoutManager=new LinearLayoutManager( context: this);
listOfEntriesAdapter=new EntryAdapter(entries);
listOfEntriesRecycle.setLayoutManager(listOfEntriesLayoutManager);
listOfEntriesRecycle.setAdapter(listOfEntriesAdapter);
```

A Bejegyzések törlése egyesével

A bejegyzések törlése az ún. „swipe” eseményre megy végbe, amikor az adott bejegyzést jobbra vagy balra elhúzzuk, akkor az adatbázisból törlődik. Ilyenkor a position alapján megkapjuk az entries listából a törlendő bejegyzést, majd a Dao delete(Entry entry) függvényével töröljük az adatbázisból, azután a listából is töröljük, majd értesítjük a rendszert a változásokról.

```
ItemTouchHelper swipeToDelete = new ItemTouchHelper(new ItemTouchHelper.SimpleCallback( dragDirs: 0, swipeDirs: ItemTouchHelper.LEFT|ItemTouchHelper.RIGHT ) {
    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder target, int direction) {
        int position = target.getAdapterPosition();
        Entry entry = entries.get(position);
        dataBase.entryDao().delete(entry);
        entries.remove(position);
         listOfEntriesAdapter.notifyDataSetChanged();
    }
});
```

Összes bejegyzés törlése

A DeleteAll gombra kattinás esetén a Dao az alábbi SQL Query-t hajtja végre: „Delete from entry” Ezzel a lista tartalmát kitörli, és egy üres listánk marad az adatbázisban, ezután az entries listát is töröljük és a változtatásokról értesítjük a rendszert.

```
private void configureDeleteAllButton(){
    Button deleteAllButton=(Button)findViewById(R.id.DeleteAllButton);
    deleteAllButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        entries.clear();
        dataBase.entryDao().deleteAll();
        listOfEntriesAdapter.notifyDataSetChanged() ;
        }
    });
}
```

GPS koordináták hozzárendelése a naplóhoz

A GPS koordináták hozzárendelését az AddNewEntryActivity-ben készítettem el, ahol a configure location függvény intézi az engedélyek elkérését és a GPS koordináták lekérését. Ezután a koordinátákat eltároljuk a létrehozott bejegyzésben

```
@RequiresApi(api = Build.VERSION_CODES.M)
private void configureLocation() {
    setGPSbutton=(Button)findViewById(R.id.setGPSCoordinates);
    coordinates=(TextView)findViewById(R.id.GPSCoordinates);
    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
    locationListener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            coordinates.append("\n "+ location.getLatitude()+" "+location.getLongitude());
        }
        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {

        }
        @Override
        public void onProviderEnabled(String provider) {

        }
        @Override
        public void onProviderDisabled(String provider) {
            startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
        }
    };
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.
            checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        requestPermissions(new String[]{
                Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.ACCESS_COARSE_LOCATION,
                 Manifest.permission.INTERNET}, requestCode: 1);
        return;
    }else{
        configureGPSButton(); }
}
```

Jelszavas belépés

A jelszavas belépés egyszerűen van megoldva, a belépés gombra kattintva a rendszer ellenőrzi, hogy a megadott jelszó azonos-e azzal, amit beállítottunk a Shared Preferencies-ben. Amennyiben nem azonos, elutasítja a belépést. Amennyiben azonban sikeres, tovább engedi a felhasználót.

Jelszóváltoztatás

A jelszóváltoztatás során a megadott új jelszót állítjuk be a Shared Perefencies-ben a „password” értékének.

```
    EditText newPasswordIn;
    Button savePassword;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_change_password);
        newPasswordIn= (EditText)findViewById(R.id.newPassword);
        configureSaveButton();
    }
    private void configureSaveButton(){
        savePassword=(Button)findViewById(R.id.addNewPassword);
        savePassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ListEntriesActivity.setPassword(newPasswordIn.getText().toString());
                finish();
            }
        });
    }
}
 public static void setPassword(String newPassword){
    MainActivity.editor.remove("password");
    MainActivity.editor.putString("password",newPassword);
    MainActivity.editor.commit();
 }
```