

|  |  |
| --- | --- |
|  |  |
| Valutaváltó projektfeladat |  |
|  |  |
|  | 20252/14 EC |
|  | Ágoston AttilaKardos ZoltánTóth Sándorné |

Tartalomjegyzék

[Valutaváltó 1](#_Toc194241966)

[2025 1](#_Toc194241967)

[2/14 EC 1](#_Toc194241968)

[Ágoston Attila 1](#_Toc194241969)

[Kardos Zoltán 1](#_Toc194241970)

[Tóth Sándorné 1](#_Toc194241971)

[A feladat célja 3](#_Toc194241972)

[A honlap témája 4](#_Toc194241973)

[Kommunikáció és tárterület management 6](#_Toc194241974)

[Az adatbázisterv 6](#_Toc194241975)

[A backend rendszer 7](#_Toc194241976)

[A fejlesztés folyamata 8](#_Toc194241977)

[Előkészités 8](#_Toc194241978)

[Projektek 8](#_Toc194241979)

[Applikáció és az admin felület összekötése 8](#_Toc194241980)

[Adatbázis 9](#_Toc194241981)

[Adatbázis láthatósága 10](#_Toc194241982)

[Adatok adatbázisból történő megjelenítése, küldése JSON formátumban 10](#_Toc194241983)

[REST API 10](#_Toc194241984)

[Serializer 11](#_Toc194241985)

[Nézetek 11](#_Toc194241986)

[urls.py beállítások 12](#_Toc194241987)

|  |  |  |  |
| --- | --- | --- | --- |
|  | A feladat célja A projektfeladatunk a Debreceni SZC Mechwart András Gépipari és Informatikai Technikum szoftvertesztelő és programozó felnőttképzési tanfolyam záró munkája.  A munka célja olyan reszponzív honlap készítése, mely képes az MNB által kezelt valuták átváltására az aktuális árfolyamon, valamint lehetőséget biztosit a regisztrált felhasználók számára valuta küldés megvalósítására. | |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | A honlap témája | A feladat kiírás úgy szólt, hogy a valós életben létező problémának informatikai eszközökkel történő megoldását kell megvalósítani. A projektmunka első fázisában ezért megegyeztünk, hogy mi egy valutaváltással foglalkozó honlapot készítünk. Ahhoz, hogy a váltás napi árfolyamon történhessen, az MNB API-járól nyerjük ki az árfolyamokat. A váltási funkciót kiegészítettük egy mini felhasználókezelési rendszerrel, valamint a bejelentkezett felhasználók számára pénzküldési felületet is biztosítunk. A harmadik megvalósult egység az árfolyamdiagrammok, mely fixen tartalmaz egy pár fontosabb valutát, de a honlap használói a rendelkezésre álló egyéb valuták közül is választhatnak megjelenítendőt.  A honlap reszponzív, igy mind asztali, mind mobileszközön megjeleníthető. |  |
|  |  | |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| A munka folyamata |
|  |

A projekten hárman dolgoztunk. Első lépésként Tóth Sándorné elkészítette a projekthez az induló adatbázist, feltöltötte az induláshoz szükséges adatokkal (alapértelmezett értékek) és ezt az adatbázist adta át a backendes munkálatokat végző Ágoston Attilának.

Attila létrehozta a backendes rendszert django segítségével. A backend rendszer részei a view-k, a serializer-ek, a felhasználókezeléshez szükséges részek, illetve néhány template, amit majd a frontend részben fognak használni. Itt történnek a számítási feladatok végzései is.

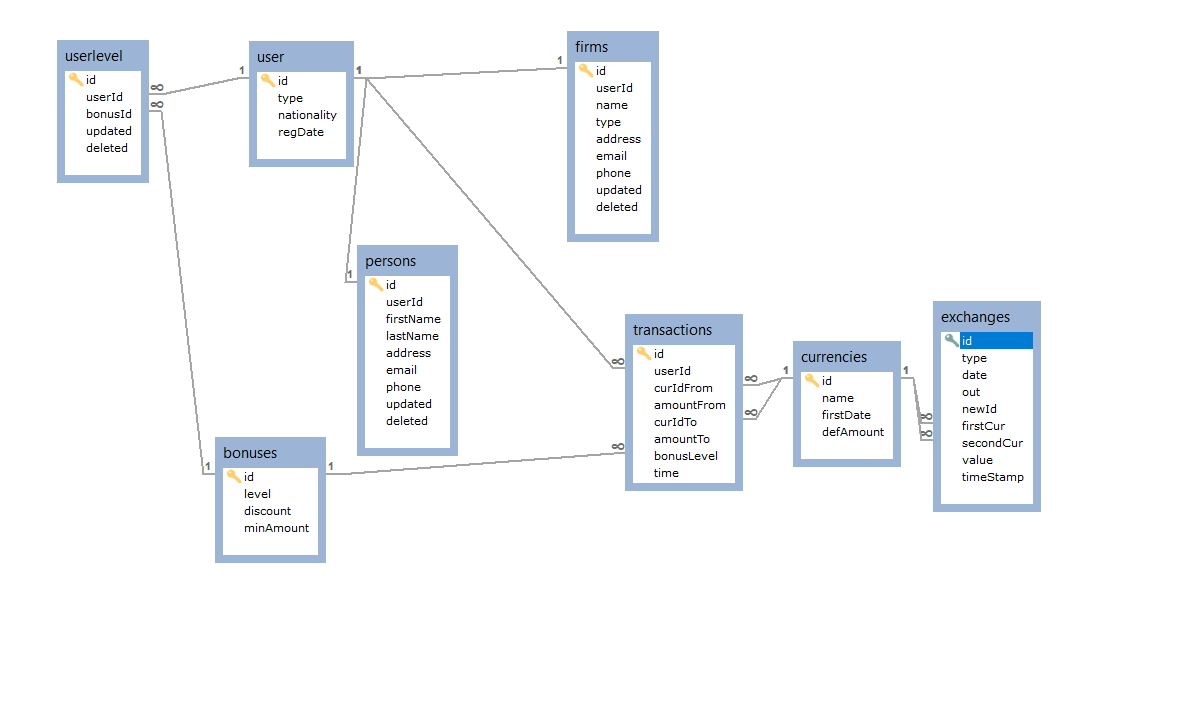
A frontend megvalósulása elsősorban Kardos Zoltán munkáját dicséri. Ő alkotta meg a honlapstruktúrát, dolgozta ki a tárhelyhasználatot, és a dizájn is az ő munkáját dicséri. A frontendhez használt alapvető módszerek a html, css, js és a bootstrap voltak.

## Kommunikáció és tárterület management

A munkafolyamat koordinálására a kedvenc eszközünk a közvetlen kapcsolat, valamint a facebook Messenger volt. A munka a GitHubon folyt, a repository-nk a [lukrecia602/project](https://github.com/lukrecia602/project) címen érhető el. A végső honlap a <http://fomix.hu/aladar/valto.html> címen található, a backend-es részek a [azenhazam.mywire.org:8800](http://azenhazam.mywire.org:8800/) szerveren futnak.

## Az adatbázisterv

Az első lépés, mint minden teljesen új informatikai fejlesztés esetében, az induló adatbázis tervezése és implementációja volt. Ezt a feladatot Tóth Sándorné végezte. Az induláskor megegyeztünk, hogy az adatokat az MNB által közzétett árfolyamokból vesszük. Igy az adatbázisstruktúrát alapvetően ezekhez igazítottuk. Mivel szükséges volt a projektben felhasználókezelést is megvalósítani, ezért ezeket az adatbáziselemek kiegészültek az ehhez szükséges elemekkel. A megvalósult adatbázis sémája az 1. ábrán található.



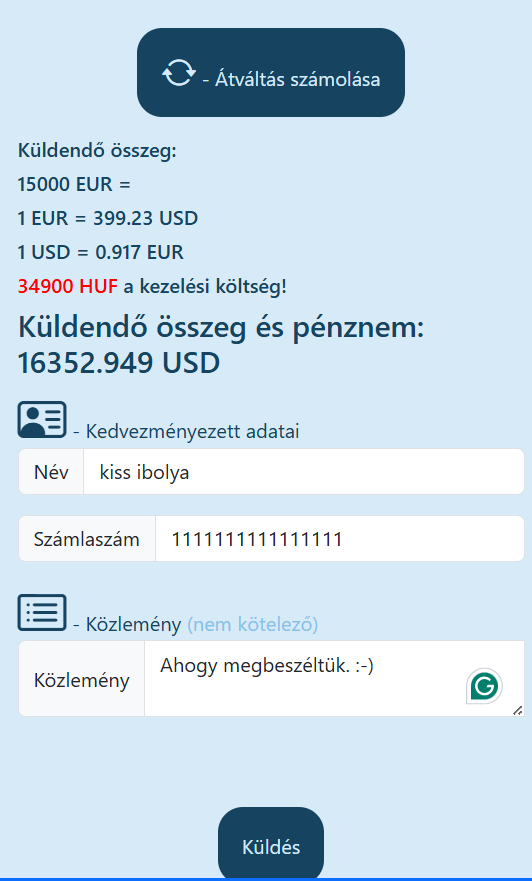
## A backend rendszer

Ágoston Attila készítette a backend egy részét django használatával. Négy template-re volt szükség, amelyek a frontendben külön html oldalt képviselnek. Ezek az oldalak alapvetően a regisztrált felhasználók által látogatott oldalak. Ezeken történik a felhasználók regisztrációja, bejelentkezése, és a valuta küldés. A felhasználók megtekinthetik a tranzakciós adataikat, törölhetik a fiókjukat. Ezekhez az oldalakhoz elkészítette a szükséges metódusokat, modelleket.

A regisztrációt, a be- és kijelentkezést, a django beépített felhasználókezelő rendszere végzi, de ennek használatához kellett készíteni egy űrlapot, amely lehetővé teszi a művelet elvégzését. Ehhez némelyik beépített metódust módosítani kellett.

Bejelentkezés után, a felhasználó tud valutát váltani majd a váltott valutát elküldeni egy címzettnek. A váltás és a küldés egy felületen történik, de külön metódusok kezelik a két műveletet.

A váltáshoz meg kell adni egy összeget, mit, mire szeretnénk váltani, a váltás gombra kattintva a program végrehajtja a váltást, és kiszámolja a küldés költségét is majd az adatok megjelennek a képernyőn.

Váltás után el tudjuk küldeni a váltott összeget. A küldéshez meg kell adni több adatot is, például a kedvezményezett nevét és számlaszámát. A küldés gombra kattintva kerülnek lementésre a tranzakciós adatok. Ezek az adatok megtekinthetők a tranzakciók gombra kattintva.Az adatok egy modell segítségével tárolódnak el az adatbázisban és egy külön metódus felel az adatok kiválogatásáért és megjelenítéséért.

## A fejlesztés folyamata

### Előkészítés

VS Codeban megnyitottam a mappát, amiben a fejlesztés fog történni.

Létrehoztam egy új parancssor típusú terminálablakot.

Ezután következett a virtuális környezet létrehozása: python -m venv .

és az aktiválása: cd Scripts\activate

Következő lépés volt a django rendszerének a telepítése: pip install django

Frissítettem a szükséges eszközöket:

python.exe -m pip install --upgrade pip

python.exe -m pip install --upgrade setuptools

Ezzel készen állt a környezet a projectek létrehozásához.

Első lépés a django adminisztrációs projektjének a létrehozása:

django-admin startproject config.

Applikáció létrehozása:

python manage.py startapp pénzváltó.

Az adatmigráció előkészítése és végrehajtása:

python manage.py makemigrations

python manage.py migrate

Admin (superuser) létrehozása:

python manage.py createsuperuser

A mi superuserünk az admin,admin adatpárral lett létrehozva, email cím nélkül.

Fejlesztői szerver elindítása:

python manage.py runserver

A szerver futását a ctrl+c billentyűkombináció szakitja meg.

Ha sikerült, akkor be lehet jelentkezni az oldalra a http://127.0.0.1:8000/admin címen.

Ha valami nem sikerült akkor a telepített projektek listázása segitett:

pip list

### Projektek

#### Applikáció és az admin felület összekötése

Az adminisztrációs projekt fájlaiban is kell beállításokat elvégezni.

- Settings.py fájlban az INSTALLED\_APPS nevű tömbben meg kell adni az app(mappa) nevét.

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'AppNeve',

]

Az urls.py fájlban össze kell kötni az Applikáció és a Adminisztráció fájljait:

from django.contrib import admin

from django.urls import path, include (includeot beirni)

urlpatterns = [

path('admin/', admin.site.urls),

path('',include('Pénzvéltó.urls'))

]

#### Adatbázis

Az appban a models.py fájlban kell létrehozni.

Különböző adattípusokkal létrehozott adatbázis:

from django.db import models

from datetime import datetime # Dátumkezelő osztály beimportálása

class Kulcs(models.Model): # Ebben a példában erre a modellre hivatkozunk idegen kulcsként.

kulcs = models.IntegerField()

def \_\_str\_\_(self):

return f'{self.kulcs}'

class Adattipusok(models.Model):

egesszam = models.IntegerField(null=False)

tortszam = models.FloatField(null=True)

datum = models.DateTimeField(default=datetime.now())

igazhamis = models.BooleanField()

rovidszoveg = models.CharField(max\_length=100)

hosszuszoveg = models.TextField()

idegenkulcs = models.ForeignKey('Kulcs', on\_delete=models.CASCADE) #idegen kulcs a másik táblára ('A másik tábla neve', CASCADE törlés)

def \_\_str\_\_(self):

return f'{self.egesszam} {self.tortszam} {self.datum} {self.igazhamis} {self.rovidszoveg} {self.hosszuszoveg} {self.idegenkulcs}'

# az admin felületen az adatok megjelenítésének beállítása: return f'{self.mezőnév} valami szöveg {self.mezőnév}'

# itt most minden adatot megjelenítünk szóközökkel elválasztva

# null = False - Azt jelenti, hogy adatfelvitelnél, az adott értéket kötelező megadni.

# null = True - Azt jelenti, hogy adatfelvitelnél az adatot nem kötelező megadni.

# default = datetime.now() - Alapértelmezett értékként a mai dátumot állítja be. Alapértelmezett érték bármilyen adattípusnál megadható.

# max\_length=100 - Maximális karakterhossz beállítása, CharField-nél kötelező megadni.

Ha kész az adatbázismodell, vagy egy már meglévő adatbázisnak változtatunk az adatszerkezetén, akkor mindig adjuk ki a

python manage.py makemigrations és python manage.py migrate parancsokat.

#### Adatbázis láthatósága

Ahhoz, hogy az adatbázis látható legyen az admin felületen, az alábbi beállítások szükségesek.

Ezeket a beállításokat az applikációbanban, az admin.py fájlban kell megcsinálni.

from django.contrib import admin

from . models import Adattipusok, Kulcs # A modellek beimportálása a modell nevének használatával.

admin.site.register(Adattipusok) # a láthatóság beállítása (Modellneve)

admin.site.register(Kulcs)

#### Adatok adatbázisból történő megjelenítése, küldése JSON formátumban

##### REST API

Ahhoz, hogy tudjunk adatokat kiküldeni JSON formátumban egy adatbázisból, több beállítás is szükséges.

Ezeket az adminisztrációs projektben a settings.py fájlban, valamint az applikációban a serializers.py, urls.py és a views.py fájlokban csináltam.

Első körben telepítettem a rest-framework-öt az alábbi paranccsal :

pip install djangorestframework.

Utána, az adminisztrációs projektben, az settings.py-ba került a következő:

INSTALLED\_APPS = [

'pénzváltóprojekt'

'rest\_framework', #rest-framework hozzáadása

]

##### Serializer

Ahhoz, hogy REST API segítégével tudjunk adatot kiküldeni JSON formátumban, szükséges egy serializer. Az appban létre kell hozni egy serializer.py fájlt és ezután ez kerül bele:

from rest\_framework import serializers

from . models import Adattipusok

class AdattipusokSerializer(serializers.ModelSerializer):

class Meta:

model=Adattipusok # Modell neve

fields = "\_\_all\_\_" # Az adatbázis összes mezőjének behúzása

depth = 1 # Mélység beállítása

##### Nézetek

A views.py fáljba kerültek azok a függvények, amik az kommunikációját és az adatok feldolgozását végzik.

Itt most a REST API-t mutatom, ami az adatok JSON formátumba történő küldéséért felel.

from django.shortcuts import render

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view

from . models import Kulcs, Adattipusok # Modellek beimportálása.

from . serializers import AdattipusokSerializer # Serializer beimportálása.

# Create your views here.

@api\_view(['GET'])

def getAllAdattipusok(request):

all = Adattipusok.objects.all() # Az adatok, tömbbe történő eltárolása.

serialized = AdattipusokSerializer(all, many=True) # Serializáció

return Response(serialized.data) # Az adatok JSON formátumban történő megjelenítése.

##### urls.py beállítások

Az applikációban hozzuk létre az urls.py-t és ezt a fájlt használjuk az adminisztrációs felület címeinek a kiterjesztésére.

from django.urls import path

from . import views # A views.py beimportálása.

urlpatterns = [

path('adatok/', views.getAllAdattipusok, name='getAllAdattipusok'. ),

]

A szerver elindítása után az adatbázisban levő adatok megjelennek a képernyőn a http://127.0.0.1:8000/adatok/ címen.