[Külső fekete borítólap formátuma]

Széchenyi István Egyetem

Gépészmérnöki, Informatikai és Villamosmérnöki Kar

Informatika Tanszék

**SZAKDOLGOZAT**

**Szűcs Henrik**

**Gazdasági Informatikus BSc szak**

2022

| [Gerincen:] Hallgató Neve, Évszám {Titkosított} |
| --- |



**SZAKDOLGOZAT**

**Webalkalmazás fejlesztése visual scripting segítségével**

**Szűcs Henrik**

**Gazdasági Informatikus BSc szak**

**2022**

**[Ennek a bekezdésnek a helyére szerkessze be az aláírásokkal ellátott feladatkiírási lap szkennelt változatát.]**

1. **Nyilatokzat**

Alulírott, *Szűcs Henrik (G28AZI)* Gazdasági Informatikus BSc szakos hallgató kijelentem, hogy a *Webalkalmazás fejlesztése visual scripting segítségével* című szakdolgozat feladat kidolgozása a saját munkám, abban csak a feltüntetett forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Győr, 2022.09.30 …………………………….

hallgató

**Kivonat**

Webalkalmazás fejlesztése visual scripting segítségével

A Construct 3 webes technológiákon alapuló visual scripting keretrendszer, melyet elsősorban játékok fejlesztéséhez terveztek. Mivel a legtöbb mai webalkalmazás a felhasználói interakciók, válaszidők és grafikai megjelenítés tekintetében is hasonló egy számítógépes játékhoz, felmerül a kérdés, hogy ezt a keretrendszert fel lehet-e használni általános célú webalkalmazások készítésére, esetleg különféle bővítmények elkészítése árán. Ez a kezdeti befektetés megtérülhet, ha később kevésbé magasan képzett fejlesztők is bevonhatók egy összetett szoftver fejlesztésének folyamataiba.

# 

**Abstract**

Web application development with visual scripting language

Construct 3 is a web based visual scripting framework which is developed for creating games. Nowadays most web applications are similar to computer games in graphics, response time and user input. The question is whether this framework can be used to build general purpose web applications even with custom made plugins. The initial extra work may pay off if fewer professional workers are enough to work on complex projects.

**Tartalomjegyzék**

[**1. Bevezetés**](#_cgzg9wnxn5k) **1**

[1.1. Témaválasztás indoklása](#_ht5770aingsc) 1

[1.2. A megrendelő problémája](#_iw62k5azmmp0) 1

[1.3. A fejlesztő problémája](#_dnaq38lyhwip) 1

[1.4. Környezet kiválasztása](#_qjxporec7zkh) 2

[1.5. Példa probléma](#_97qb6khpvas) 6

[**2. Szerkesztő bemutatása**](#_m4n5yksiwj3g) **8**

[2.1. Telepítés](#_gwp0mgjvyx21) 8

[2.2. Beállítások és eszközök:](#_ln7d56ja7m61) 8

[2.3. Új Projekt](#_a57087y5d6mn) 9

[2.4. Projekt szerkezete](#_ya3xefsku10z) 9

[2.5. Kezelőfelület beállítása](#_cfw0esac7rv3) 11

[2.6. Példa elkészítése](#_eacerrrqxfp) 13

[2.6.1. Projekt beállítása](#_vu88l3gzef4) 13

[2.6.2. Elrendezés](#_9g6o739kim8u) 14

[2.6.3. Esemény lap](#_47ntf6nv40t1) 17

[2.6.4. Exportálás](#_intp6c9mo38z) 23

[**3. Tömeges email küldő**](#_9j711pr2wm6p) **24**

[3.1. Működés](#_p0ultmilryc9) 24

[3.2. Felület](#_800k7l2wr4os) 25

[2.3. Hiányzó elemek](#_68kyeh8ztt0l) 26

[**4. Kiegészítők elkészítése**](#_5oz68eun4o0t) **27**

[4.1. Fejlesztői mód](#_ak0snspfdk8s) 27

[4.2. JavaScript környezet](#_84hjieh06xqr) 28

[4.3. Lokalizáció](#_xlytzhtjpzio) 29

[4.4. Titkosítás](#_a0zoeb1lzdj2) 30

[4.5. Asztali operációs rendszer](#_aodg6kcm5kng) 30

[4.6. Fájlkezelés](#_6s7nemiczsqs) 30

[4.7. Email küldés](#_umr99ok43hf1) 30

[4.8. Szövegszerkesztés](#_gd5krzs9ncvv) 30

[**5. Tömeges email küldő elkészítése**](#_19ggyqyhmpj6) **30**

[**6. Konklúzió**](#_gsszlbn8111u) **31**

[**Irodalomjegyzék**](#_wsikuiwoz7qw) **32**

[**Mellékletek**](#_4cdc7r4caisp) **35**

# 1. Bevezetés

## 1.1. Témaválasztás indoklása

Munkám során a megrendelők egyedi programokat rendelnek, azonban azok átadását követően általában felmerül a program bővítésének igénye. A hagyományos módszerekkel elkészített programokat csak egy speciális szaktudású programozó tudja bővíteni. Személyes célom, hogy egy megírt programot könnyen lehessen bővíteni és a programkészítés folyamata bárki számára elérhető legyen még akkor is, ha nem jártas az adott programozási nyelvben. A szakdolgozat egy problémán keresztül mutatja be az elképzelésem megvalósítását. A cél, hogy a kialakított szerkesztő környezet támogatni tudja a felhasználók kreativitását. Eszközt szeretnék adni az embereknek ahhoz, hogy saját programokat tudjanak létrehozni és a megírt programot hatékonyan tudják csapatban, másokkal együtt továbbfejleszteni.

## 1.2. A megrendelő problémája

A szoftverfejlesztés egyik legnagyobb nehézsége, hogy a megrendelőnek egyedi igényei lesznek. A szoftvert az adott feladathoz testre kell szabni és a megfelelő informatikai környezetbe kell illeszteni, ami a nagy költségekkel jár. Az egyedi szoftverfejlesztés költségei nagyon magasak már a munka kezdetekor. Külön szakembert/csapatot kell bérelni és az esetleges további módosításokhoz is a szakemberhez/csapathoz kell fordulnunk. Ha a prototípus elkészítése valamilyen okból nem sikerül, az nagy veszteségeket okozhat, mert ennek kifejlesztése során is szakembereket kellett fizetnünk és elkészült program sem jött létre.

## 1.3. A fejlesztő problémája

Problémát jelent a fejlesztések költségének féken tartása, illetve a szükséges szakemberek keresése, betanítása. A költségek nagy részét a munkaerő bére teszi ki. Gazdaságosabb a megrendelt programot nem teljesen alapjaitól megírni, érdemes kiválasztani egy keretrendszert (sablont), hogy akár kevésbé szakképzett fejlesztő, kevesebb idő alatt készítsen el egy programot. Továbbá a fenntartási költségeken is érdemes spórolni, ha az alkalmazás funkcióit önállóan, külső szolgáltatás igénybevétele nélkül is tudjuk üzemeltetni.

## 1.4. Környezet kiválasztása

Keretrendszer kiválasztásakor a legfontosabb szempontok lehetnek az alábbiak:

* **platformfüggetlen**: Az elkészített programok működjenek ugyanúgy a különféle rendszereken. Lehetőleg csak megkerülhetetlen platformspecifikus megkötések legyenek (Pl.: Asztali környezeti tálca ikon, áruház integráció stb.). További előny, ha a fejlesztőkörnyezet is platformfüggetlenül használható.
* **bővíthető**: A keretrendszer funkcióit ki lehessen egészíteni saját tartalmakkal, kódokkal. Előny, ha a kiegészítés teljesen vagy részlegesen kompatibilis egyéb programnyelvvel, illetve programokban használt kiegészítőkkel/tartalmakkal, ezen felül támogatást biztosít ezek importálásához.
* **könnyen kezelhető**: A keretrendszerhez legyen támogatás, dokumentáció és frissítés, ami a későbbi munkát segíti. A fejlesztői program kezelése könnyen elsajátítható legyen (pl. hasonló programoktól kevés eltérés). A fejlesztői vagy az elkészített program lehetőleg „Out of the Box” működjön, vagyis beüzemelése ne igényeljen más külső, nem mellékelt keretrendszer(eke)t, extra beállításokat, illetve összetett konfigurációt (Pl.: önállóan futtatható legyen, hordozható legyen a program, ne igényeljen külső SDK letöltést, ne kelljen környezeti változókat, regisztereket beállítani stb.).

A piacon rengeteg rendelkezésre álló programnyelv és szoftver van. Keresésnél elsődleges szempontom, hogy asztali, mobilos és webes környezetben is működjön az elkészített program, mert személyes tapasztalatom, hogy a megrendelők sokféle eszközön akarják majd futtatni a megrendelt applikációt. Másodlagos szempontom, hogy lehetőség legyen önállóan futtatható (monolit) program készítésére, így készíthető szolgáltatás független alkalmazás, ami további költség megtakarítást jelent, mert nem kell szolgáltatást fenntartani, karbantartani.

Számos programozási nyelven íródnak programfejlesztési eszközök, ezért nincs programnyelvbeli megkötésem, de számomra a webes technológiák ismerete miatt választáskor a JavaScript-re épülő rendszerek élveznek előnyt.

* **Flutter** [1]: Modern és reszponzív felhasználó felület építésére összpontosító keretrendszer, azonban itt a közös kódot a fordító platformspecifikus natív kóddá alakítja, így a platformok között eltérés tapasztalható.
* **React** [2]: Ingyenes és nyílt forráskódú JavaScript keretrendszer, kliens oldali alkalmazásokhoz. A Meta cég és a felhasználói közössége tartja karban. A keretrendszer elsősorban a felhasználói felület készítését segíti. Ehhez programozói tudás és fejlesztői környezet telepítése, beállítása szükséges.
* **Vue** [3]: Ingyenes és nyílt forráskódú JavaScript keretrendszer, kliens oldali „egylapos” alkalmazásokhoz. Funkcióját tekintve hasonló a React-hez. Megjelenítés HTML elemeken keresztül történik, a különböző elemeket interaktívan kezeli.

Számtalan további keretrendszer létezik, de ezek a keretrendszerek speciális programozói tudást igényelnek, telepítésük, a kód karbantartása, továbbadása és verziókezelése nehézkes lehet egy kezdő számára, így egyszerűbb keretrendszereket kerestem. CMS (Content Management System) rendszerek a tartalom kezelésére összpontosítanak, általában kényelmes grafikus felületet biztosítanak a működés és tartalom kezelésére.

* **Wordpress** [4]: Eredetileg blogger keretrendszernek készítették. Webszerverre történő telepítés után a felhasználó webes böngészőn keresztül testre szabhatja weboldalát grafikus kezelőfelületen. Kiegészítőkkel bővíthető, azonban a legnagyobb hátránya, hogy kevés jól működő alapfunkciója van. A keretrendszert használó pedig eltévedhet a rengeteg félkész, fizetős, kis funkcionalitást biztosító kiegészítők tengerében.

Számtalan további tartalomkezelő rendszer létezik különböző célokra optimalizálva, pl. Joomola[5], Drupal[6], Wix[7]. A bővíthetőség, könnyen kezelhetőség adottak, de kizárólag webszerverre telepítve működik, webes környezetben fut, offline futtatható alkalmazás készítésére nem alkalmasak és a kiegészítők összességében drágává tehetik.

Programok készítésére „no code”/„low code” keretrendszerek is rendelkezésre állnak. Ezek a rendszerek a programkészítés során a hagyományos kódolási feladatokat csökkentik és áthelyezik egy kezdők számára könnyen átlátható grafikus felületre. Hasonló grafikus felületet biztosítanak, mint a CMS rendszerek, de ezeknél maga a program készítés nagyobb hangsúlyt kap.

* **Appsheet** [8]: Google üzleti alkalmazás készítő szolgáltatása. Többi Google szolgáltatás összekötésével egyedi munkamenetünkhöz igazodó alkalmazást hozhatunk létre.
  + Előnyök: Jól támogatott, egyszerű a használata
  + Hátrányok: Csak szolgáltatás formában elérhető, nem lehet teljes mértékben testre szabni.
* **Appery.io** [9]: Webes és platformspecifikus applikáció készítő szolgáltatás. Vizuális szerkesztő mellett backend szolgáltatásokat is biztosít. Különböző bővítményekkel bővíthető az alkalmazás.
  + Előnyök: Sokféle funkció és kiegészítés, nagyfokú testre szabási lehetőség
  + Hátrányok: Csak szolgáltatás formában elérhető, kisebb projektekhez drága

Legtöbb ilyen rendszer csak szolgáltatásként érhető el és a teljeskörű bővítése nem lehetséges. Rengeteg hasonló további „no code” [10] és „low code” [11] alkalmazás létezik a piacon.

Tovább bővítettem a keresést játékmotorokra is, mivel ezekkel a rendszerekkel olyan programokat lehet írni, amelyek külső szolgáltatás nélkül, gyorsan tudnak reagálni a felhasználói bevitelre, animációkat, rendszer funkciókat tudnak kezelni és önállóan is futtathatók.

* **Unity** [12], **Unreal Engine** [13], **Godot** [14]: Ezeknek a motoroknak az alapelve, hogy stabil, jól bevált nyelveken, (C, C++, C#, Python stb.) fejlett funkciókat valósítanak meg, amelyeket a programozó felhasználhat. Mivel a felhasznált nyelvek legtöbb funkcióját kihasználja a motor, a fejlesztőnek is haladó tudással kell rendelkeznie az adott programozási nyelven és a motorról.
  + Előnyök: Nagy közösség, kiterjedt funkcionalitás, fejlett szerkesztő
  + Hátrányok: Használata speciális tudást igényel (programozási nyelv, függvények/motor ismerete)
* **Construct 3** [15], **Game Maker: Studio** [16], **Gdevelop** [17]: Ezen programok a fentiekben bemutatott programok ötvözete. Gyors tanulási lehetőséget biztosítanak vizuális programozási lehetőséggel (visual scripting), azonban lehetőség van a motort bővíteni hagyományos programozási nyelvekkel, így bonyolultabb projektek készítésére is alkalmas.
  + Előnyök: Bárki számára könnyen elsajátítható, bővíthető
  + Hátrányok: kevésbé professzionális megoldások, esetleges sebesség béli korlátok

A kész program költségcsökkentése céljából olyan keretrendszerek közül választok, amelyek nem igényelnek különleges képzést és tudást, így a munkaerő költsége, illetve a kész program elkészítési ideje is csökkenthet. Végül a **Construct 3**, **Game Maker: Studio**, **Gdevelop** programokra szűkítem a keretrendszer választásomat, de az alábbiakban a lehető legjobb választási eredmény érdekében a programokat egymással is összehasonlítom.

**Game Maker: Studio**

C++-ban írt fizetős asztali játékprogram szerkesztő. Úgy a vizuális programozást (GML Visual), mint a szerkesztő saját programozási nyelvét is támogatja (GML Code). A szerkesztő saját GML fordítót használ a kész program C++ kódjának elkészítéséhez. Ennek fizetős modellje változatos funkciókat támogat.

**Gdevelop**:

JavaScriptben írt webes szerkesztő. Az exportált program is JavaScriptben készül egy környezeti motorba beágyazva. A program teljesen ingyenes. Csak vizuális szerkesztés lehetséges. Kiegészítők készítésére is lehetőség van, azonban haladóbb, programozói szinten nem lehet bővíteni ezt a programot. A program felépítése kissé összefüggéstelen (nehéz átlátni, hogy mely objektumokból hogyan épül fel a program).

**Construct 3**:

JavaScriptben írt fizetős webes szerkesztő program. Az exportált program is JavaScriptben készül egy környezeti motorba beágyazva. Vizuális programkódot és a JavaScript programozási nyelvet is támogatja. Kiegészítők készítésére is lehetőség van JavaScript nyelven. Nagy hangsúly van a generált kód optimalizálásán, modern API-okat, illetve technológiákat használ.

**Game Maker: Studio** – **Gdevelop** összehasonlítása:

„Game Maker: Studio” fejlettebb C++-ban írt motorral, jobb funkciókkal, megvalósítással és támogatással rendelkezik. A Gdevelop viszont ingyenes és beüzemelése is egyszerűbb. „Game Maker: Studio” bővíthetősége jobb, mint a Gdevelop programnak. Az egyszerűbb beüzemelés és használat miatt, én mégis a Gdevelopot részesítem előnyben.

*1. táblázat: Game Maker: Studio, Gdevelop összehasonlítása*

|  | Game Maker: Studio | Gdevelop |
| --- | --- | --- |
| Funkciók | Fejlettebb | Kevésbé fejlett |
| Használat | Nehezebb | Egyszerűbb |
| Ára | Fizetős (1550 Ft/hó) | Ingyenes |

**Construct 3** – **Game Maker: Studio** összehasonlítása:

„Game Maker: Studio” első látásra fejlettebbnek tűnik, azonban Construct 3-ban az optimalizáció hangsúlyosabb [18]. A Construct 3 egyszerűbben elérhető, könnyebben kezelhető, bővíthetősége rugalmasabb. Dokumentáció és SDK jobban elérhető Construct 3-hoz.

*2. táblázat: Construct, Game Maker: Studio összehasonlítása*

|  | Game Maker: Studio | Construct 3 |
| --- | --- | --- |
| Funkciók | Kevésbé fejlett | Fejlettebb |
| Használat | Nehezebb | Egyszerűbb |
| Ára | Fizetős (1550 Ft/hó) | Fizetős (3240 Ft/hó) |

**Construct 3** – **Gdevelop** összehasonlítása:

A Gdevelop programot a Construct 3 ingyenes „kis testvérének” nevezném. Mindkettő JavaScript kódot generál, visual scripting alapú a programlogika. A Gdevelop ingyenességéből adódóan több hátrány is keletkezik: funkciók implementáláshoz költséghatékonyabb és rosszabb minőségű programokkal rendelkezik (motor sebessége is elmarad a Construct 3-hoz képest). A Construct 3-hoz minden keretrendszert egybecsomagoltak és beágyaztak, így nem szükséges további rendszerek telepítése.

3. táblázat: *Construct, Gdevelop összehasonlítása*

|  | Gdevelop | Construct 3 |
| --- | --- | --- |
| Funkciók | Kevésbé fejlett | Fejlettebb |
| Használat | Nehezebb | Egyszerűbb |
| Ára | Ingyenes | Fizetős (1550 Ft/hó) |

Személyes választásom a Construct 3-ra esett. Fizetős a program, de az általam támasztott követelményeknek ez felel meg a legjobban. Kiegészítőkkel bővíthető, visual scripting segítségével könnyen használható, a generált JavaScript kód platformfüggetlenül, önállóan futtatható és elég gyors.

## 1.5. Példa probléma

Small/home office, kisvállalati, oktatási környezetben jellemző, hogy évente egy-két alkalommal rengeteg felhasználónak kell köremail-t küldeni. Valós példák: Országos Magyar Vadász Kamaránál az éves jegy hosszabbítás február végén történik és az esetleges tudnivalókról több ezer tagot kell értesíteni egyidőben. Az oktatásban a tanároknak évkezdéskor több száz diákot és szülőt kell értesíteniük a változásokról köremail formájában.

A megrendelők egy olyan szoftveres megoldást szeretnének, amivel akár 1000-2000 email is elküldhető. Külön szervert és informatikust nem tudnak biztosítani a beüzemelésre. Lehetséges megoldások a problémára:

* Szolgáltatás igénybevétele: Egy tömeges email küldő szolgáltatásra történő előfizetés (Mailchimp, Mailgun, stb.).
  + Előnyök: Azonnal rendelkezésre áll, nagy kínálat a szolgáltatások között
  + Hátrányok: Szolgáltatás mivolta miatt az egyszeri használat drága lehet (általában a cél a nagyobb csomag megvásárlása).
* Program használata (Outlook, Thunderbird, stb.):
  + Előnyök: teljesen független az előfizetéstől, bárki számára hozzáférhető, használható. Hosszú távon olcsóbb.
  + Hátrányok: email fiók szükséges

A megrendelők inkább egy számítógépen futó programot képzelnek el, mint például az Outlook, Office, Chrome stb. Jelenleg a következő programokat találtam alkalmasnak a tömeges email küldésre:

* Thunderbird: Mozilla alkotóinak levelező kliense. Nyílt forráskódú, ingyenes, rengeteg platformra letölthető. Ingyenessége miatt nagyrészt közössége fejleszti. Nincsenek prémium funkciók és szolgáltatások a programhoz mellékelve. Használata kicsit archaikus, gyakorlást igényel, ha másik levelező rendszerről térünk át használatára.
  + Előnyök: ingyenes
  + Hátrányok: tömeges levélküldés csak „3rd party” (külsős) kiegészítővel lehetséges, ennek nehézkes a beüzemelése (kiegészítők megbízhatósága kérdéses)
* Sendblaster: Delivery Tech Corp. által fejlesztett Windows-os szoftver (megjelenéséből ítélve valószínüleg .NET, WPF applikáció), a fejlesztő által „befejezett” szoftvernek látszik. Ára 99 euró + a frissítések és kiegészítők további költséggel járnak.
  + Előnyök: egyszerű használat, kész alkalmazás
  + Hátrányok: drága (a fő verziókat külön kell megvenni), csak Windows rendszerre elérhető, jövőbeli frissítései kérdésesek (2020.04.10 utolsó frissítés)

Megrendelő a „Sendblaster” szoftverhez hasonló programot szeretne. Cél, hogy a szerkesztővel készített program a megrendelő egyedi igényeit megvalósítsa és a fejlesztő gazdaságosan, fenntarthatóan tudja elkészíteni a programot.

# 2. Szerkesztő bemutatása

A szerkesztő felépítését és szerkesztőben történő programszerkesztést egy alap példa programon keresztül fogom bemutatni. A példa feladatban egy adott téglalap területét kell kiszámítani. A program a téglalap két oldalának hosszát fogja kérni és kiírja a kiszámolt téglalap területét.

## 2.1. Telepítés

A szerkesztő bármely verziójának megnyitásakor a böngésző elkezdi letölteni és frissíteni a különböző forrásokat. A letöltés végeztével a program internet nélkül is használható, azonban a felhőbe való mentéshez, felhőbe történő exportáláshoz és a licenckulcs frissítéséhez internetkapcsolat szükséges.

Frissítések heti rendszerességgel érkeznek. Szakdolgozat írásakor a r317 volt a legfrissebb stabil verzió. A szerkesztőt különböző verziókban, módokon lehet elindítani:

* Normál / stabil verzió: <https://editor.construct.net/>. Stabil kiadás az azt megelőző béta ciklus után (8-10 verzió) érkezik. A stabil verziók a béta frissítéseket is tartalmazzák.
* Béta verzió: <https://editor.construct.net/beta>. Új funkciók, komplexebb javítások első megjelenése. Kevésbé tesztelt, néha hibásan működhet.
* Megadott verzió betöltése: https://editor.construct.net/rXYZ pl.: 308.2-es verzió <https://editor.construct.net/r308-2>. Az adott verzió betöltése hosszabb távú projekteknél lehetővé teszi a frissítések kihagyását, így hibamentesebb fejlesztés kivitelezhető.
* Biztonságos mód: <https://editor.construct.net/?safe-mode>. Minden „3rd party” (külső fél) kiegészítőjét kikapcsolja, hogy elindulhasson a szerkesztő, ha valamelyik kiegészítő hibásan működött.

## 2.2. Beállítások és eszközök:

Bármely verzió indításakor egy előre beállított kezdőképernyő fogadja a felhasználót, ahol hasznos linkeket, ajánlott minta projekteket és a legutóbb megnyitott projekteket érheti el.



*1. ábra: Construct szerkesztő kezdőoldala*

A felső sávon a balra lévő *Menu* (*Menü*) gomb állandó elem. Ennek segítségével hívhatók elő a szerkesztő fontosabb beállításai.

*Menu* (*Menü*) > *Settings* (*Beállítások*) menüpont tartalmazza a szerkesztő beállításait. Itt beállíthatók a nyelv, felületi beállítások, mentési helyek és szerkesztői konfigurációk.

Szerkesztőt kiegészítőkkel lehet bővíteni a *Menu* (*Menü*) > *View* (*Nézet*) > *Addon manager* (*Kiegészítő kezelő*) menüpontban. Itt lehet a telepített kiegészítőket megtekinteni, törölni, illetve új kiegészítőket telepíteni.

## 2.3. Új Projekt

Új projekt létrehozásához a *Menu* (*Menü*) > *Project* (*Projekt*) > *New* (*Új*) elemet kell használni. Projekt létrehozásakor egy felugró ablakban kezdeti konfigurációkat adhatunk meg, ezek a beállítások később is módosíthatók. Jelenleg semmilyen beállítást nem adunk meg, csak a *Create* (*Létrehozás*) gombra kattintunk.

Létrehozás után plusz ablakok jelennek meg a kezelőfelületen. Ezek testre szabásáról a későbbiekben lesz szó.

## 2.4. Projekt szerkezete

Projekt létrehozása után alapértelmezett beállítások használatakor a jobb oldalon a *Project* (*Projekt*) ablakban látható a projekt szerkezete:



*2. ábra: Szerkesztő projekt ablaka*

Projekt nevével ellátott mappa a főkönyvtár. Ezen belül almappák, az almappák-on belül elemek és/vagy további almappák találhatók. A mappák és elemek átnevezését, törlését, másolását és egyéb funkciókat tartalmazó menüt a jobb egérgombbal lehet előhívni. A mappák funkció szerint vannak kialakítva:

* **Layout (Elrendezések)** mappa tartalmazza az elrendezéseket, vagyis azokat a felületeket, melyeket a felhasználó a kész programban látni fog. Elrendezésen belül egymás alatt elhelyezkedő rétegek találhatók. Egyszerre egy elrendezés jeleníthető meg, de a rétegeket egy másik elrendezés módosíthatja. Elrendezésekhez egy-egy eseménylap is társítható.
* **Event sheets (Esemény lapok)** mappa tartalmazza az esemény lapokat, amelyek az elrendezések vezérlését végzik. Az eseménylap fogja tartalmazni a program konkrét működési logikáját. Az eseménylapok egymásba is beágyazhatók.
* **Scripts (Szkriptek)** mappa JavaScript kódok tárolására szolgál. Az itt tárolt programkódoknak ugyanaz a feladata, mint az esemény lapoknak, csak nem vizuálisan történik a programlogika elkészítése, hanem JavaScript nyelven egy szöveges fájlban.
* **Object types (Objektum típusok)** mappa tartalmazza a projekt építő elemeit, az objektumokat. Minden látható vagy nem látható elemet, funkciót (Gomb, Hang, Videó, Adattárolás) objektumok valósítanak meg, ezeket a projekthez kell majd hozzáadni.
* **Families (Családok)** mappában családok szerepelnek. Ezek olyan csoportosítások, amelyekbe az azonos típusú objektumokat (pl.: Gombokat) egy csoportba (családba) rakhatjuk, ezzel egyszerűbben, egy egészként tudjuk őket kezelni (pl.: a családban lévő minden szöveg objektumot egyszerre piros színre tudunk állítani)
* **Timelines (Idővonalak)** mappában idővonal elemek találhatók. Egy idővonal elemmel a különböző objektumok értékeinek időbeli vezérlését adhatjuk meg. *Eases* (*Átmenetek*) almappában az egyedi átmenetek találhatók. Az átmenet két értékváltozás közötti interpolációt [20] (pl.: lineáris, köbös stb.) adja meg.
* **Sounds (Hangok)** mappa olyan hangfájlok tárolására szolgál, amelyeknek az időben történő lejátszása kritikusan fontos (pl. kattintás hangja), ezért a mappába másolt hangokat a program indítás előtt be fogja tölteni.
* **Music (Zenék)** hasonlóan a hang mappához itt is hangfájlokat lehet tárolni, de az itt található audió fájlok nem töltődnek be a program indulásával, így lehetnek kisebb akadozások a lejátszás során.
* **Videos (Videók)** mappában videó fájlok tárhelye.
* **Fonts (Betűtípusok)** különböző betűtípus fájlok (.ttf, .woff, stb.) tárhelye.
* **Icons (Ikonok)** a program által használt ikon fájlok (.png). Beállítható, hogy fájl melyik ikon típushoz legyen társítva.
* **Files (Fájlok)** mappában bármilyen típusú fájl tárolható, akár az előzőekben kategorizált fájltípusok is. Csak az egyéb, nem kategorizálható fájlokat érdemes ide másolni, mert az ebben a mappában lévő fájlokat a szerkesztő nem fogja tudni optimalizálni (pl.: hang esetében nem végez tömörítést).

## 2.5. Kezelőfelület beállítása

A *Menu* (*Menü*) > *View* (*Nézet*) > *Bars* (*Sávok*) menüből nyolc különböző ablakot lehet előhívni, ezek az ablakok különböző funkciókat biztosítanak [19]. A projekt elrendezéseit, esemény lapjait, egyéb fájljait is külön ablakban lehet megjeleníteni.



*3. ábra: Szerkesztő alapértelmezett felülete*

Minden ablak törlése után, csak a felső menü elem marad.



*4. ábra: Szerkesztő üres felülete*

Az ablak megnyitása után, az ablak szabadon mozgatható az adott felületen az ablak felső szélének egérrel történő húzásával.

Lehetőség van az ablakok illesztésére is, ez az ablak kék négyzetekre történő ráhelyezésével tehető meg. Illeszteni a képernyő vagy egy ablak négy oldalára lehet. Ha ráillesztjük az ablakot egy másik ablakra, akkor a megjelenő fülekkel lehet váltani az egymást fedő ablakok között. Az illesztés helyét jelzi a program.



*5. ábra: Szerkesztő felülete átrendezés közben*

Mobilon történő megjelenítés esetén az idővonal ablak nem elérhető, illetve az ablakok nem változtathatók, csak oldalról húzhatók be.

A továbbiakban a példa készítésénél az alapértelmezett elrendezést fogom használni, vagyis bal oldalon a *Properties* (*Tulajdonságok*) ablak, jobb oldal felső részén a *Project* (*Projekt*) ablak, jobb alsó részén pedig a *Layers* (*Rétegek*) ablak jelenik meg.

## 2.6. Példa elkészítése

### 2.6.1. Projekt beállítása

Egész projektre vonatkozó beállítások eléréséhez kattintsunk bal egérgombbal a projekt nevére a *Project* (*Projekt*) ablakban. Ezután a tulajdonság ablak a projekt tulajdonságait fogja megmutatni. *Properties* (*Tulajdonságok*) ablakban szerkesszük a következő értékeket:

* *Name* (*Név*) mezőbe írjuk be a projekt nevét: „Terület számoló”
* *Description* (*Leírás*) mezőben adjunk meg: „Téglalap terület számoló program.”
* *ID* mezőbe írjuk be: „com.teglalap.terulet”
* *Author* (*Szerző*) mezőbe írjuk be saját nevünket.
* *First layout* (*Első elrendezés*) listából válasszuk ki az egyetlen elrendezésünket.



*6. ábra: Projekt tulajdonságainak beállítása*

*Viewport size* (*Nézetablak méret*) mezőben a programunk felbontását állíthatjuk be. A többi beállítással jelenleg nem foglalkozok, a projekt létrehozásakor a sablonok ezeket a projekt tulajdonságokat előre beállítják.

### 2.6.2. Elrendezés

*Project* (*Projekt*) ablak > *Layouts* (*Elrendezések*) mappa > *Layout 1* (*Elrendezés 1*) elemet dupla bal egér kattintással vagy jobb egérgomb és *Open* (*Megnyitás*) gombbal nyithatjuk meg (ha még nem lenne megnyitva). Ilyenkor az elrendezés nevével elnevezett ablakban lesz látható az elrendezés.

Ha kiválasztjuk az elrendezés elemet egyszeri bal egér kattintással, akkor a *Properties* (*Tulajdonságok*) ablakban megjelennek az elrendezés tulajdonságai. Szerkesszük a *Size* (*Méret*) mezőket 854x480-as értékre, ezzel az elrendezésünk méretét a projektünk virtuális méretéhez igazítottuk.



*7. ábra: Elrendezés tulajdonságainak beállítása*

Projektünk különböző részeit objektumok fogják alkotni. Ezeket az *Object types* (*Objektum típusok*) mappában tudjuk kezelni. Új objektum hozzáadásához először a *Project* (*Projekt*) ablakban > *Object types* (*Objektum típusok*) mappára > jobb egérgombbal kattintva > *Add new object type* (*Új objektum típus hozzáadása*) gombbal kell megnyitni a *Plugins* (*Bővítmények*) listáját.



*8. ábra: Objektum hozzáadása*

A példa programunkhoz két beviteli (hosszúság és szélesség) mező és egy megjelenítési (kiszámolt terület) mező szükséges. A beviteli mezők *Text Input* (*Beviteli mező*) típusú objektumok lesznek, míg az eredmény megjelenítéséhez egy *Text* (*Szöveg*) típusú objektumot használunk.



*9. ábra: Objektum lista*

A létrehozott három objektumot egérrel történő húzással elhelyezhetjük az elrendezésen. Rendezzük el tetszés szerint a szövegdobozokat.

A bal egér kattintással kijelölt objektumok tulajdonságai meg fognak jelenni a *Properties* (*Tulajdonságok*) ablakban. Állítsuk be a három objektum tulajdonságát:

* Eredmény szöveg:
  + *Name* (*Név*) legyen: „Eredmeny”
  + *Text* (*Szöveg*) legyen „Eredmény:”
* Hosszúság beviteli mező:
  + *Name* (*Név*) legyen: „Hosszusag”
  + *Placeholder* (*Helyőrző*) legyen „Hosszúság”
* Szélesség beviteli mező:
  + *Name* (*Név*) legyen: „Szelesseg”
  + *Placeholder* (*Helyőrző*) legyen „Szélesség”



*10. ábra: Elrendezésen lévő objektumok szerkesztése*

Ezzel a felhasználói felület elkészült. *Menu* (*Menü*) > *Project* (*Projekt*) > *Preview* (*Előnézet*) gombbal vagy fenti lejátszás gombbal kipróbálható a projekt. A különböző eszközökön egyformán fog megjelenni az elkészített elrendezésünk (jelenleg két beviteli mező és egy szöveg).

### 2.6.3. Esemény lap

A programot esemény lapokban lévő utasításokkal vezérelhetjük. Az eseménylap ugyanolyan funkciót tölt be, mint egy programkód, csak a megírt vezérlés emberközelibb formában jelenik meg pl.: leíró jellegű, külön részekre bontható, nincs szigorú szintaxis, beépített „varázslók” segítenek (végigvezetnek a beállításokon).

*Project* (*Projekt*) ablak > *Event sheets* (*Esemény lapok*) mappa > *Event Sheet 1* (*Esemény lap 1*) elemet dupla bal egér kattintással vagy jobb egérgomb és *Open* (*Megnyitás*) gombbal nyithatjuk meg (ha nem lenne még megnyitva). Ilyenkor az esemény lap nevével elnevezett ablakban lesz látható az esemény lap.



*11. ábra: Üres eseménylap szerkesztése*

Egy esemény lap Action (Művelet), Condition (Feltétel), Expression (Függvény) (angol rövidítés: „ACE”) típusú blokkokból épül fel. A blokkok hierarchikusan épülnek fel, egy blokk tartalmazhat feltételt, műveletet, illetve további alblokkokat. Típusai az alábbiak lehetnek:

* Művelet a program „cselekvése”. Valamilyen konkrét műveletet végez pl.: írás, olvasás, áthelyezés, megjelenítés stb. Két fajta művelet létezik:
  + **Szinkron művelet**, olyan művelet aminek a befelyezését meg kell várni, tovább művelet nem futtatható közben.
  + **Aszinkron művelet**, ahol a művelet kezdete és befejezése között folytatható a további műveletek futtatása. Ezt olyan funkcióknál alkalmazzák ahol a kezdés és befejezés között hosszabb idő telik el pl.: fájl beolvasása, webről történő adatkérés stb. A szerkesztő egy óra szimbólummal jelöli.
* Feltételek feladata: a műveletek lefutásának kezelése. A feltételben lévő blokkok akkor aktiválódnak, ha a feltétel igaz. A feltételeknek 3 fajtája van:
  + **Normál feltétel**, akkor aktiválja az alatta elhelyezkedő blokkokat, ha a feltétel igaz.
  + **Ciklus feltétel** hasonló a normál feltételhez, de a feltétel újra ismétli a benne lévő blokkokat, amíg a feltétel igaz. A szerkesztő egy ismétlés ikonnal jelzi ezt.
  + **Esemény** olyan feltétel, ami egy történés hatására (pl.: egérkattintás, betöltés befejezése stb.) lefuttatja a blokkot. A szerkesztő jobbra mutató zöld nyíllal jelöli ezt.
* **Függvény** segít a műveletek és feltételek paramétereinek megadását. A függvényeknek szöveg vagy szám formájában paramétereket adhatunk és szöveget vagy szám értéket kapunk eredményként.

Az esemény lap logikája (kivéve az eseményeket) minden képernyő frissítésekor lefut (általában 60-szor másodpercenként).

Most a konkrét példát készítjük el, vagyis ha a felhasználó megadja a hosszúságot és szélességet, akkor a program beolvassa a két oldalt és összeszorozza, az eredményt pedig kiírja a képernyőre. Példa programunkban egy eseménnyel figyeljük azt, hogy a beviteli (hosszúság és szélesség) mező tartalma megváltozott-e, vagyis a felhasználó beírt-e valamit.

1. *Add event* (*Esemény hozzáadása*) gombbal megnyitjuk a varázslót. Kiválasztjuk a hosszúság megadására szolgáló korábban megadott szöveg beviteli objektumot (nálam „Hosszusag” névvel rendelkezik).
2. A felsorolt feltételek közül kiválasztjuk az *On text changed* (*Szöveg változásakor*) esemény feltételt. Ez további paramétert nem kér, így hozzá is adja a szerkesztő az esemény lapunkhoz.

Mivel a másik beviteli mezőt (hosszúságot) is figyelni kell, ehhez is létrehozunk a fenti módszerrel egy eseményt. A két esemény összefűzhető egy feltétel blokkba, azonban a feltétel blokk alapértelmezetten *AND* (*ÉS*) kapcsolatot akar kialakítani a két feltétel között. Ezt meg kell változtatnunk *OR* (*VAGY*) kapcsolatra. A blokk szélére jobb egér kattintással előhívható menüből keressük ki a *Make OR block* (*VAGY blokk létrehozása*) elemet.



*12. ábra: Feltétel blokk VAGY kapcsolatra állítása*

A VAGY blokkban már összekapcsolható a két feltétel. Ebben a blokkban lévő további blokkok akkor fognak aktiválódni, ha a hosszúság vagy szélesség mezőt megváltoztatja a felhasználó.

Most a meglévő feltételek alá szeretnék új üres blokkot létrehozni. Ezt úgy tudjuk megtenni, hogy a létrehozott feltétel bal szélére (így az egész feltétel blokkot kijelöli, nem csak a feltétel részeket) jobb egér gombbal kattintunk, majd az előhívott menüből az *Add blank sub-event* (*További üres alesemény hozzáadása*) elemre kattintunk.



*13. ábra: Üres blokk hozzáadássa*

A létrehozott üres blokkhoz adjunk egy változót is, amiben a kiszámolt területet tároljuk. Ezt az üres blokkra történő jobb egérkattintással, majd az *Add* (*Hozzáadás*) > *Add local variable* (*Lokális változó hozzáadása*) gombbal tehetjük meg. A felugró ablakban adjuk meg a változó nevét („Terulet” nevet fogom használni).



*14. ábra: Változó hozzáadása*

Most kiszámoljuk a területet az oldalak összeszorzásával majd eltároljuk azt az előbb létrehozott változóban. Az üres blokk mellett lévő *Add action* (*Művelet hozzáadása*) gomb megnyomásával előugró ablakban válasszuk ki a *System* (*Rendszer*) elemet, mert a változókkal történő műveletek rendszer objektumban találhatók. *Set value* (*Érték beállítása*) művelet kiválasztása után a következő párbeszédablakban a művelet paramétereit állíthatjuk be. Nem kell más változót megadni (összesen egy változónk van a projektben), az érték részhez pedig a területet kell megadni.

A függvények segítségével dinamikusan tudjuk számolni az oldalakból a területet. A *Find Expressions* (*Kifejezések keresése*) gomb segítségével előhívható, hogy az adott objektumoknak milyen függvényei vannak, illetve milyen és hány paramétert kell átadni a függvényeknek. A következő függvényeket fogjuk használni:

* Beviteli mező *Text* függvényét, ami a beviteli mezőbe írt aktuális szöveget adja vissza. Hivatkozni a beviteli mező nevével lehet, majd ponttal elválasztva tudjuk a megfelelő függvényt használni pl.: Hosszusag.Text, Szelesseg.Text
* Rendszer *float* függvényét, amely az első paraméterben megadott szöveget konvertálja számmá, ha az nem konvertálható, nullát ad vissza.

A területszámítási összefüggés a következőképpen alakul: float(Hosszusag.Text) \* float(Szelesseg.Text)



*15. ábra: Területszámítás elkészítése*

A területszámítás elkészítése után ellenőrizzük, hogy a terület nagyobb-e, mint nulla. Ha igen, akkor kiírjuk az eredményt képernyőre, egyébként kiírjuk, hogy nem jók az adatok.

Adjunk a területszámítás blokkjának alsó szomszédságába egy üres blokkot (fenti üres blokk hozzáadása rész ismétlése). Az újonnan létrehozott üres blokkra duplán bal egér gombbal kattintva feltételt adhatunk meg. *System* (*Rendszer*) objektumon belül *Compare variable* (*Változó hasonlítása*) feltételt válasszuk ki. A paramétereket úgy állítsuk be, hogy ha a változó nagyobb mint nulla, akkor legyen igaz a feltétel.



*17. ábra: Területszámítás ellenőrzése*

A feltételhez adjunk meg kiíratási műveletet. *Add action* (*Művelet hozzáadása*) gomb után válasszuk ki az eredményt megjelenítő szöveg objektumot. A szöveg objektum *Set text* (*Szöveg beállítása*) művelet segítségével kiírható az eredmény. Statikus szöveget “-jel között lehet megadni, &-jellel lehet összefűzni a szövegeket.

****

*18. ábra: Eredmény kiírása*

A különben feltételt az egész feltételre történő bal egérkattintással, majd *Add* (*Hozzáadás*) > *Add 'else'* (*'Különben' hozzáadása*) gombbal tehetjük meg. A létrejött feltételhez készítsünk egy műveletet, ami az eredmény szövegbe kiírja, hogy „hibás eredmény” (Lokalizációt nem támogat a szerkesztő ezért ezt hasonlóan kell megtenni, mint az eredmény kiírást, csak statikus „hibás eredmény” szöveggel). Az elkészült programunkat tesztelhetjük is a fenti lejátszás ikonnal.



*19. ábra: Végső programlogika*

### 2.6.4. Exportálás

Exportálás során az elkészült projektünket önállóan futtatható programmá tesszük. Asztali exportáláshoz menjünk a: *Menu* (*Menü*) > *Project* (*Projekt*) > *Export* (*Exportálás*) > *NW.js* elemre.



*20. ábra: Projekt és végső program kapcsolata*



*21. ábra: Exportálás elkezdése*

A felugró ablakon lévő beállítások maradhatnak alapértelmezett értékeken, de a *Minify mode* (*Tömörítési mód*) elemet *advance* (*haladó*) értékre állíthatjuk gyorsabb, kisebb méretű programhoz, azonban ebben a módban lassúbb lesz az exportálás.

A következő ablakban az operációs rendszert és egyéb NW.js-re vonatkozó paramétereket választhatjuk ki (alapértelmezett beállítások megfelelőek). Ha valamely rendszerre nem akarunk exportálni, akkor az operációs rendszer neve melletti jelölőnégyzet ne legyen kijelölve a felsorolásban.

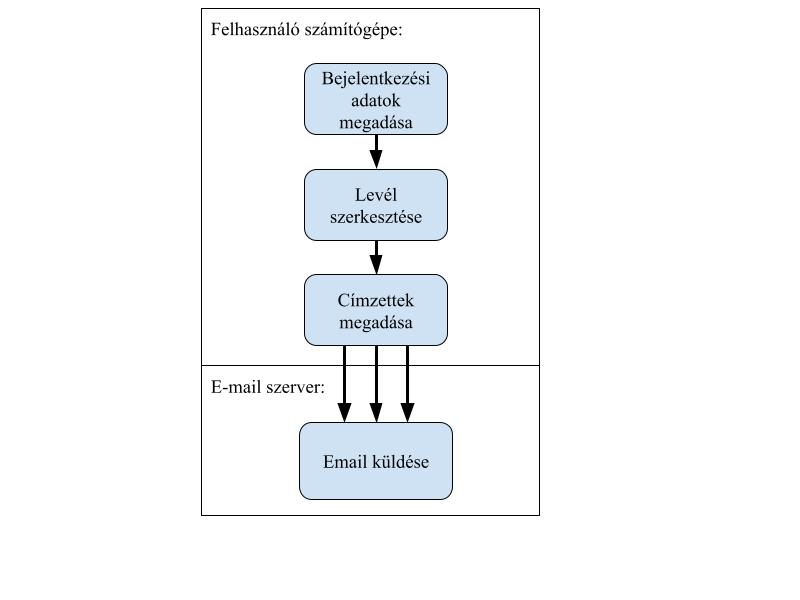
Ezután elkezdődik a program exportálása. Exportálás után egy csomagolt fájlt fog letölteni a böngészőnk. Ez a fájl a programunk futtatható változatát tartalmazza.

# 3. Tömeges email küldő

## 3.1. Működés

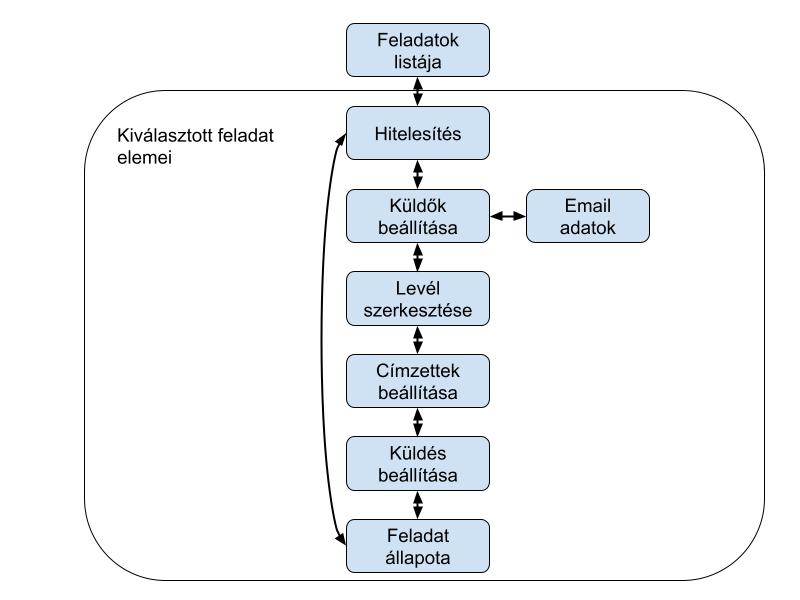
Az email küldő applikáció hasonlítani fog egy hagyományos levelező kliensre. A felhasználónak a meglévő e-mail címének bejelentkezési adatait meg kell adni, majd a program utasítja a levelező szervert SMTP protokollon keresztül az e-mailek küldésére.

Tömeges e-mail küldés esetén a levelek egyszerre történő küldése nem működőképes a címzettek nagy száma miatt. A leveleket egyesével vagy kisebb csoportokban kell elküldeni a levelező szerver felé. Gyorsabb e-mail küldés érdekében több e-címről, levelező szerverről is küldhetők lesznek a levelek. A programban beállítható lesz a küldés idejének és sebességének megadása is.



*22. ábra: Tömeges email küldés folyamata*

## 3.2. Felület



*22. ábra: UI folyamatábra*

* **Feladat lista**: A kezdőlapon az e-mail küldési feladatok és állapotaik láthatók. Ezen a felületen kiválasztható egy e-mail küldési feladat vagy elkészíthető egy új is. A feladat kiválasztása után menüpontokon keresztül állíthatjuk be a különböző beállításokat.
* **Hitelesítés**: Ezen a felületen át lehet nevezni a feladatot, illetve új feladat esetén jelszó állítható be, meglévő feladat esetén jelszó megadása kötelező a folytatáshoz.
* **Küldők beállítása**: A küldést végző e-mail címek listáját lehet itt beállítani. Egy darab e-mail cím mindenképpen szükséges. Az e-mail adatainak kezelése vagy hozzáadása egy külön felületen történik. Az e-mail beállító felület az e-mail cím SMTP szerver helyét, titkosítási módját, felhasználó nevét és jelszavát fogja kérni. Lesznek sablonok, amelyekkel könnyebb lesz az adatok beírása.
* **Levél szerkesztése**: Ebben a felületben lehetőség van a levél tartalmának szerkesztésére. Levelet egy egyszerűsített szövegszerkesztőben lehet szerkeszteni.
* **Címzettek beállítása**: Címzettek listája vesszővel elválasztott értékekkel (CSV) készíthető. A fejléc opció is bekapcsolható lesz, így a levélben személyre szabott paramétereket lehet beállítani pl.: név, titulus stb.
* **Küldés beállítása**: Küldés sebessége és időpontja állítható be.
* **Feladat állapota**: Ezen a felületen lesz látható a feladat állapota (jelenleg fut-e vagy befejezett-e), naplózása, továbbá ezen a felületen indítható és állítható le a folyamat.

## 2.3. Hiányzó elemek

A program elkészítéséhez a szerkesztő nem biztosít minden funkciót.

* lokalizáció,
* titkosítás,
* asztali rendszereken háttérben történő futás,
* fájlkezelés,
* e-mail küldés,
* szövegformázás,

funkciók teljesen hiányoznak a szerkesztőből, ezen funkciókat fogom megvalósítani a szakdolgozat keretein belül. Felhasználói felületek létrehozásához a lehetőségek limitáltak a Construct 3-ban, azonban ez nem akadályozza a program fejlesztését. A felhasználói felületek létrehozása a projekt számára még elégséges, így nem fókuszálok a felhasználói felületek bővítésére.

# 4. Kiegészítők elkészítése

A szerkesztőhöz különböző kiegészítőket telepíthetünk. A jelenlegi projekt hiányzó funkcióit nem közvetlen a programra szabva fejlesztem ki, hanem általánosabb funkciókat készítek kiegészítők formájában. Ily módon ezek a funkciók más programban is használhatók lesznek.

## 4.1. Fejlesztői mód

Ahhoz, hogy fejlesztés alatt álló kiegészítőket tudjunk telepíteni a szerkesztőhöz, be kell kapcsolnunk a Construct fejlesztői módját. A *Menu* (*Menü*) > *Settings* (*Beállítások*) ablak felső szélére történő négy darab bal egér kattintással megjeleníthetjük az *Enable developer mode* (*Fejlesztői mód engedélyezése*) jelölőnégyzetet, ezt bejelölve a fejlesztői módot bekapcsolásra kerül. A fejlesztői módban telepített kiegészítők minden indításkor újra betöltődnek, így a legutolsó módosítás is azonnal tesztelhető.



23. Ábra: Fejlesztői mód bekapcsolása

A biztonsági szabályok miatt fejlesztői bővítményeket csak a „https://localhost/” url címekről lehet betölteni. Jelenleg egy Electron környezetben megírt és előre konfigurált webszervert (EDS - Easy Development Server) használok a fejlesztői bővítmények könnyebb betöltéséhez, de használható Xampp vagy egyéb más webszerver is.

Az Easy Development Server egy titkosított HTTP szervert futtat gyorsítótárazás nélkül. A programban a következő négy mező állítható be:

* Bővítmény útvonala: kiválasztható a bővítményt tartalmazó mappa, ezt a mappát fogja a program elérhetővé tenni a „https://localhost/” url-en.
* Port: megadja azt, hogy localhost cím melyik portján lehessen elérni a mappát
* Privát kulcs: titkosításhoz használt privát kulcs
* Tanúsítvány: titkosítás hitelesítéshez használt tanúsítvány. Ezt az operációs rendszerre külön is fel kell telepíteni, hogy a titkosítást biztonságosnak ítélje meg a böngésző.

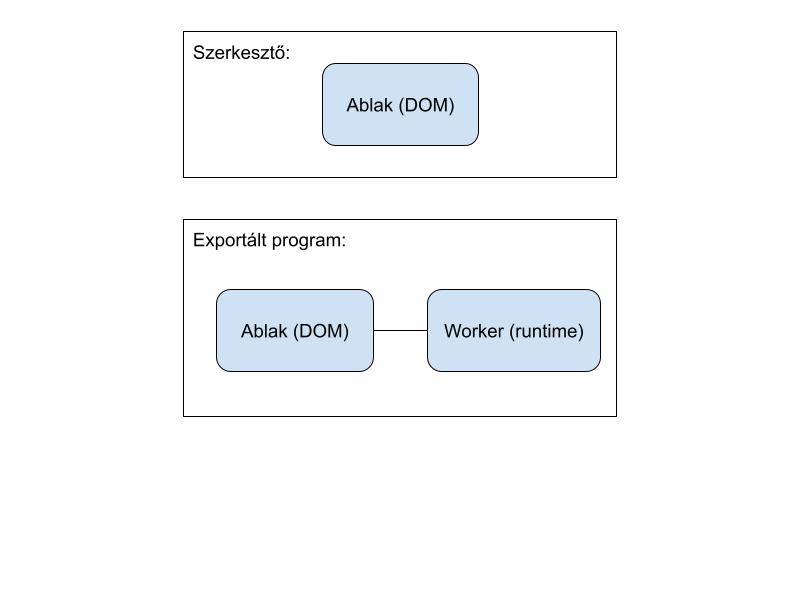
Start gomb megnyomása után a program kiír egy url-t, ahol a kiegészítő elérhető.



24. Ábra: Easy Development Server működés közben

## 4.2. JavaScript környezet

JavaScript kód különböző környezetekben fut. A szerkesztőben futtatott JavaScript kód hozzáfér a „window” elemhez, viszont az elkészített program kódja külön szálon fog futni. Az exportált programban a fő szál és a megnyitott szálak egy csatornán kommunikálnak egymással.



*24. Ábra: JavaScript kontextus a szerkesztőben és az exportált programban*

Az elkészített JavaScript kódnak támogatnia kell a Google Closure Compiler által biztosított egyszerű és összetett tömörítést is [xx].

Vannak egyéb Construct direktívák a JavaScript használatra és formázására:

* Szigorú mód használata („use strict”).
* Hozzáférés (scope) korlátozása kapcsos zárójelekkel: {}
* Deklaráláskor const, let használata
* Osztály metódusok elnevezése szóközök nélkül, szavak nagy kezdőbetűvel: OsztalyMetodus
* Belső, saját használatú osztály metódusok alsó kötőjel kezdéssel: \_OsztalyMetodus
* Javascript metódusok nevénél első betű kicsi, egybe, szavak nagy kezdőbetűvel: peldaValtozo
* Hordozható objektumok kulcs nevei kisbetűvel, kötőjellel elválasztottak legyenek: pelda-valtozo
* Nem tisztázott esetekben a Google Stílus Útmutatója az irányadó (<https://google.github.io/styleguide/jsguide.html>)

## 4.3. Lokalizáció

A programok különböző nyelveken való megjelenítése manapság alapkövetelmény. Különböző nyelvek betöltéséhez a legegyszerűbb módszer, az ha a szövegeket egy kulcs-érték párban tároljuk és megjelenítéskor a megfelelő értékeket kiolvassuk. Ez azonban sok szöveggel rendelkező program esetén nehézkesen kezelhető, a különböző programrészek nehezen kapcsolhatók össze. Constructban részleges megoldást kínálnak a külső felek által készített [x2] kiegészítők. Ezek kevés rugalmasságot biztosítanak, nyelv felismerés és kezelés sincs implementálva (kizárólag a betöltést könnyítik meg). A saját kiegészítőm szöveget, nyelvet és paraméterezett szövegeket fog kezelni, továbbá különböző tárolási módszerek is támogat.

A kiegészítő a következő tárolási formátumokat fogja támogatni:

* JSON (single): JSON fájl ahol a végső érték a fordítás szövege. Betöltéskor meg kell adni a forrás nyelvét.
* JSON (multiple): JSON fájl ahol a végső érték egy olyan objektum, ahol nyelvenként fel vannak sorolva a különböző fordítások.
* CSV: szabványos vesszővel elválasztott értékek, ahol az első sor(fejléc) azonosítja a különböző nyelvek fordításait.
* Construct 3 szótár: szerkesztőben létrehozott szótár fájl. Betöltéskor meg kell adni a forrás nyelvét.
* Construct 3 tömb: szerkesztőben létrehozott tömb fájl, ahol a fejléc (első sor) azonosítja a különböző nyelvek fordításait.

## 4.4. Titkosítás

A kiegészítő feladata különböző titkosítási algoritmusok alapján adatokat titkosítani, továbbá a titkosítást kiegészítő műveletek elvégzése (ellenőrző összeg számítása, kulcsgenerálás stb.). Elsősorban a JavaScript kriptográfiai könyvtárát illesztem a Construct 3-hoz [xx3].

## 4.5. Asztali operációs rendszer

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

## 4.6. Fájlkezelés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

## 4.7. Email küldés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

## 4.8. Szövegszerkesztés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# 5. Tömeges email küldő elkészítése

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# 6. Konklúzió

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fringilla molestie tellus non dignissim. Quisque congue enim non erat gravida venenatis.

# Irodalomjegyzék

[1] Flutter

<https://docs.flutter.dev/resources/architectural-overview>, 2022.11.18 16:22

[2] React

<https://reactjs.org/>, 2022.11.18 16:22

[3] Vue

<https://vuejs.org/>, 2022.11.18 16:22

[4] Wordpress

<https://hu.wordpress.org/about/>, 2022.11.18 16:22

[5] Joomola

<https://www.joomla.org/about-joomla.html>, 2022.10.25 22:03

[6] Drupal

<https://www.drupal.org/about>, 2022.10.25 22:03

[7] Wix

<https://www.wix.com/about/us>, 2022.10.25 22:03

[8] Appsheet

<https://about.appsheet.com/home/>, 2022.11.18 16:22

[9] Appery.io

<https://appery.io/>, 2022.11.18 16:22

[10] List of low-code development platforms

<https://en.wikipedia.org/wiki/List_of_low-code_development_platforms>, 2022.11.18 16:22

[11] List of no-code development platforms

<https://en.wikipedia.org/wiki/No-code_development_platform>, 2022.11.18 16:22

[12] Unity

<https://unity.com/>, 2022.11.18 16:22

[13] Unreal Engine

<https://www.unrealengine.com/en-US/>, 2022.11.18 16:22

[14] Godot

<https://godotengine.org/>, 2022.11.18 16:22

[15] Construct 3

<https://www.construct.net/en>, 2022.11.18 16:22

[16] Game Maker: Studio

<https://gamemaker.io/en/gamemaker>, 2022.11.18 16:22

[17] Gdevelop

<https://gdevelop.io/features>, 2022.11.18 16:22

[18] How JavaScript beats GameMaker Language (GML)

<https://www.construct.net/en/blogs/construct-official-blog-1/javascript-beats-gamemaker-1570>, 2022.10.18 17:22

[19] Construct’s interface

<https://www.construct.net/en/make-games/manuals/construct-3/overview/the-interface>, 2022.11.18 16:22

[20] Interpoláció

<https://hu.wikipedia.org/wiki/Interpol%C3%A1ci%C3%B3>, 2022.11.18 16:22

[xx] Példa

<https://example.com/>, 2022.11.18 16:22

[A felhasznált szakirodalom megadása]

[A szerzők nevét mindenütt “Családnév, X.” formában kell megadni, ahol X. a szerző keresztnevének (keresztneveinek) kezdőbetűje. Magyar cikk esetén a vessző a családnév és a keresztnév kezdőbetűje közt elhagyható. Ha az egyértelműség megkívánja, a keresztnév kiírható teljesen is.

A dolgozat szerzője szabadon választhat, az A vagy a B típust használja.]

**[A-típus:**

A cikkekre való hivatkozás egy []-be írt sorszámmal történik. A sorszámozást folytonosan kell megtenni, a sorba rendezés alapja az első szerző családneve.]

[1] Szerző1 (, Szerző2 ...): *Cikk címe*

Folyóirat neve, sorszám, kezdőoldal-végoldal, év.

[2] Szerző1 (, Szerző2 ...): *Konferencia-kiadvány-beli cikk címe*

„Konferenciakiadvány:” Konferencia neve, hely, kezdőoldal-végoldal, év.

[3] Szerző1 (, Szerző2 ...): *Könyvcím*

„Könyv:” Kiadó, hely, oldalszám, év.

[4] Szerző1 (, Szerző2 ...): *Kutatási jelentés címe (csak publikus elérhető jelentés!)*

„Kutatási jelentés”: Kutatási projekt neve, intézet, oldal, év.

[5] Szerző: *Disszertáció címe*

„PhD/kandidátusi/stb. disszertáció”: Egyetem, kar neve, év.

[6] *Internetes oldal elnevezése:*

*URL, letöltés ideje (csak konkrét dokumentumra mutató URL adható meg!)*

**[B-típus:**

A hivatkozás a szerzők családi nevéből és a kiadás évéből képezett azonosítóval történik. Kettőnél több szerző esetén az „et.al.” rövidítés használható. Pl. „[Vijayasundaram, 1986.]”, „[Meister and Sonar, 1998.]”, „[Felcman et.al., 1994.]”

Abban a ritka esetben, amikor több cikknek is azonos azonosító jutna (megegyeznek a szerzők és a kiadás éve) az év után az „a”, „b”, „c”, stb. betűk csatolandók, pl. „[Stone and Norman, 1993a.]”.

A sorba rendezés alapja a szerzők családneve, végül az év.

Ezt leszámítva a formátum ugyanaz, mint az A-típus esetén, de ekkor a hivatkozási sorszám lehagyható.]

# Mellékletek

1. [A dolgozat mellékletei, ha vannak]