**Helsinki feladat**

**Készítette:**

* Oláh Gergő
* Vasilescu András
* Gombos Benedek

[Bevezetés 4](#_Toc1188822957)

[Tervezés 5](#_Toc53122580)

[Megvalósítás 5](#_Toc1440904569)

[Tesztelés (táblázat) 6](#_Toc731021803)

[Felmerülő problémák 8](#_Toc248612401)

[Összegzés 9](#_Toc1120074170)

[Felhasznált irodalom 10](#_Toc1817964004)

# **Bevezetés**

A csoportunk következő feladata egy Helsinki nevezetű oldal elkészítése volt. Ez a feladat eredetileg egy vizsga feladat volt Csharp-ban, ahol mind forms és mind console-os részét megkellett csinálni, viszont nekünk egy teljesen más feladat adódott. Készítenünk kell egy oldalt, ahol kiiratjuk a Helsinki feladat szöveges állományát egy menüben, ezzel alkalmazva a forms tudásunkat csak JS-ben.

A feladat, ahogy említettük kezdetben csak Csharp-ban kellet csinálni, de jelenleg JS, HTML tudásunkat kell majd előtérbe helyezni és hasznosítani.

CSS-t is elkell készíteni és a HTML-t is, illetve a vizsga feladat részeit is el kell készíteni, mivel manapság a console-os programoknak nincs nagy jelentősége, itt ugrik be a HTML rész, hogy akkor készítünk hozzá egy oldalt, ami igazából a CSS feladat csak más kódban megírva.

Összeségében, egy oldalt kell készítenünk, ahol a forms tudást átkell rakni stílus kódba. mivel a kettő majdnem teljesen egyezik, így az ott tanultak átvihetőek az oldalba.

Úgy gondoljuk, hogy ez a feladat könnyebb, mint az előző, amit elkészítettünk, de úgy érezzük ebből is csak tanulni tudunk majd, illetve megértjük a kódok közötti különbséget is és azonosságokat.

# **Tervezés**

A tervezést a Csharp-ban kezdtük, ahol a forms-ot tervezzük elkészíteni, abból kiindulva, hogy a feladat Csharp és HTML is, így azt terveztük, hogy a forms-ot elkészítjük először és azt átalakítjuk vagy importáljuk HTML-be, mivel a két kód szinte egyezik így egyszerűbb a könnyebb verzióban megírni és a nehezebbe átrakni véleményünk szerint.

A kiosztás hasonló volt az előző projektünkhöz, a kódokat együtt írjuk meg, de főképp 2 ember dolgozik rajta legtöbbet a másik 1 személy meg vizsgálja és néhol ránéz, hogy kell-e segítség vagy netán jól halad-e a munka és ha baj van kisegíti őket.

A dokumentáció meg a tesztdokumentáció ugyanúgy a munka közben készül, mivel így látjuk reálisabbnak, hogyha program futása közben írjuk így egyszerűbben tudunk haladni.

A közös munkához kódban a LiveShare nevezetű kiegészítőt alkalmaztuk újra, ennek lehetőségével képesek vagyunk remote-ban dolgozni és ha távol is vagyunk képesek vagyunk egyszerűbben dolgozni, illetve láthatjuk azt, hogy bármelyikünk mit is javított/írt, ha egyedül dolgozott későbbi időben.

Így a tervet mondhatni egy hosszú folyamatra terveztük. Először a forms megcsinálása Csharp-ban, másodszor a console rész elkészítése, harmadszor a forms-ban elkészített „alap” be importálása a HTML kódba, negyedszer a CSS elkészítése a Csharp-ban elkészített forms igazából, szóval mondhatni átmásolása a kódnak és kész lesz az a része is, végére pedig jön majd a lezárása az egésznek, simítgatások, finomítások stb.

# **Megvalósítás**

A megvalósítást először a Csharp forms részén kezdtük, mivel ott akartuk megalkotni az „alapot”, ahogy a tervben is ecsetelgettük. A forms-ot 3-an kezdtük el csinálni, mivel gyorsan akartunk haladni vele, de végül mindenkinek teljesen más lett és úgy döntöttünk inkább csinálja egy ember, mert haladni is tudtunk úgy, illetve reálisabb, ha egy ember dolgozik és a többi be segít, ha kell. Andrisé lett a JavaScript és a weboldal, Benedeké a Csharp, és a forms része a programnak és Gergő csinálja a teszt-, illetve a teljes dokumentációt. Igaz, mindenki külön-külön dolgozik, de együtt készítjük el mindegyik feladatot részletben.

1. **Csharp**

Benedek írta a Csharpot, ahogy említettük korábban.

Legeljén a forms applikációval kezdte a feladatot, ahol kezdetben megírta az alap menü részlegét és utána kezdte el a feladatok többi részét.

Elején közösen készítettük el az 1-5-ig terjedő feladatokat, és illetve 8-ba is belesegítettünk, mivel azok voltak a jelentősebb és néhol nehezebb részei a feladatnak, végül kész lett a menüsor és hozzá tartozó részek, feladatok és elvárások is.

1. **JavaScript**

Andris írta a weboldalt, ahol besegítettünk neki az alapot elkészítésénél, illetve az egy-két feladatnál is. A javát magától csinálta meg, és az alap koncepció az volt, hogy az oldalt aszerint csinálja meg, hogy rá lehessen külön keresni az adatokra, nem volt feladat része, de egy kis bónusznak beleraktuk.

Feladatokat és elvárásokat elkészítettük és az oldal is tökéletesen funkcionált.

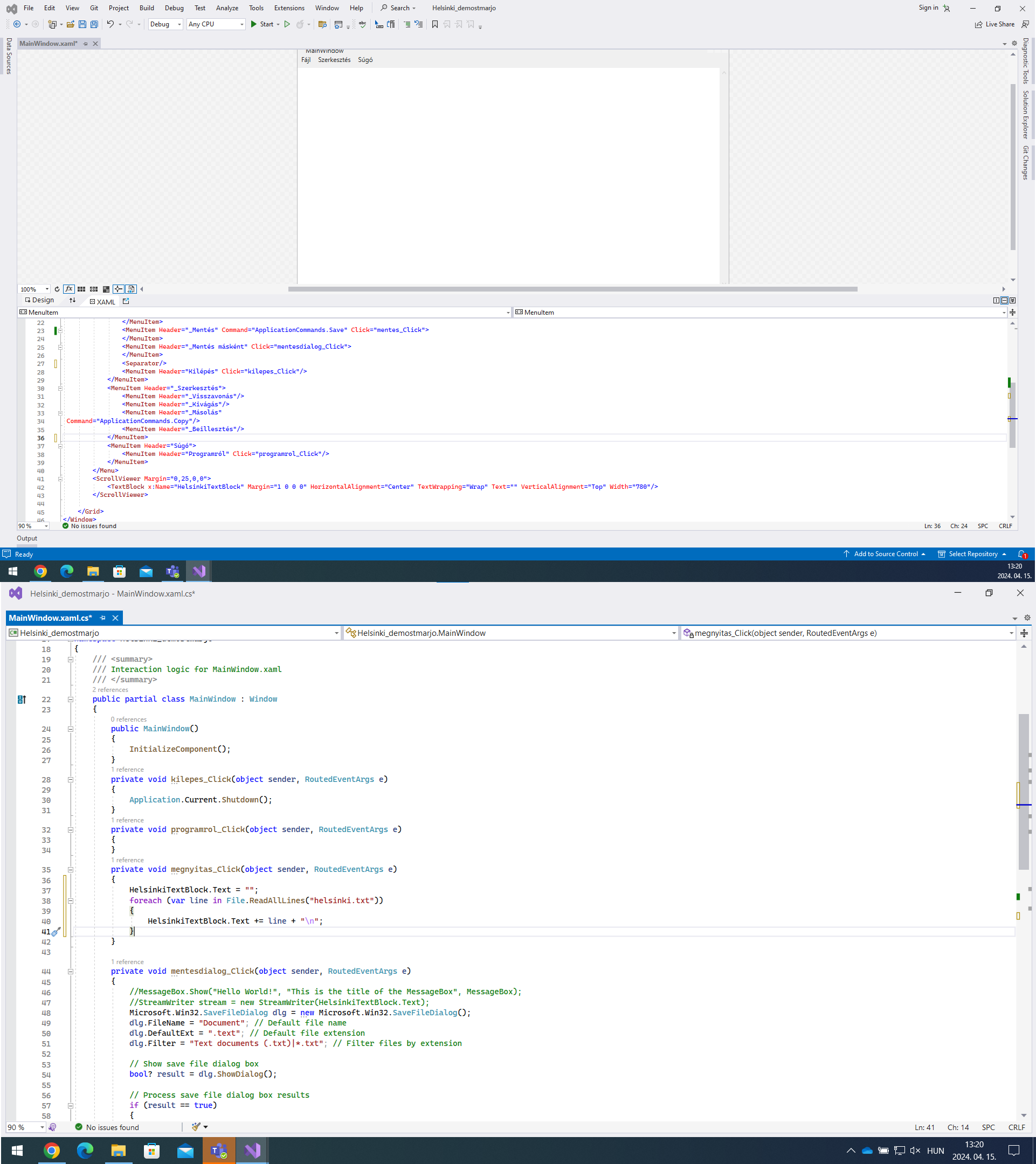
1. **Dokumentáció**

A dokumentációt főképp együtt csináltuk, de csak a tesztelés részét, az alap teljes dokumentáció részét Gergő csinálta meg, a tesztdokumentációt András és Benedek kódírás közben kezdték el vezetni és rakták bele a megadott hibákat és ez a táblázat végül be lett rakva a dokumentációba külön fülként.

A dokumentációt külön részekre szedtük, hogy átláthatóbb legyen és hogy minden egyes részre kibírjunk térni, amely, lényegesnek bizonyult.

# Tesztelés (táblázat)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Date** | **Tester** | **Test Case** | **Test Step Desc** | **Error value/description** | **Expected Result** | **Outcome** | **Solving Date/Method** |
| 2024.03 | Benedek | WPF Menü működése | Menü megjelenik és a lenyíló rész működik | - | Menü rendesen funkcionál | Sikeres | - |
| 2024.03 | Gergő | WPF Megnyitás | Megnyitás fül beolvassa a fájlt | - | Megnyitja a fájlt és beolvassa a gombbal | Sikeres | - |
| 2024.03 | Gergő | WPF Megnyitás és beolvasás | Beolvasva a fájlt gombnyomásra csak egyszer | A program többszörösen olvasta be és nem állt meg egynél | Csak egyszer olvassa be a fájlt | Hibás | - |
| 2024.03 | András | WPF Megnyitás és beolvasás | Beolvassa a fájlt gombnyomásra csak egyszer | - | Csak egyszer olvassa be a fájlt | Sikeres | - |
| 2024.03 | Benedek | JS  Fájl beolvasása és kiiratása | Beolvassuk a fájlt és az oldal indításakor a console-ban megjelenik a tartalma | - | Beolvasás és a kiiratása teljesüljön | Sikeres | - |
| 2024.03 | András | JS  Adatok darabokra szedése, kiiratása | Adatokat darabokra szedjük, tömbbe rakjuk és kiiratjuk a képernyőre | - | Az adatok táblázatba felsorolva jelenjenek meg | Sikeres | - |
| 2024.03 | Gergő | JS  Keresőmotor működése | Valamelyik sportra rákeresve kiadja az adott sportban résztvevők adatait | - | Keresésnél történik egy szortír | Sikeres | - |
| 2024.04 | András | JS  Adatok listába kimutatása | Az adatok megjelennek felsorolva beolvasáskor | 404 error, nem találja a fájlt | A tömb adatai kiíródik az oldalra | Hibás | - |
| 2024.04 | András | JS  Adatok listába kimutatása | Az adatok megjelennek felsorolva beolvasáskor | - | A tömb adatai kiíródik az oldalra | Sikeres | A fájl megadott helye, ahonnan lekérve volt, nem volt helyes |



# **Felmerülő problémák**

Felmerülő problémának értjük azokat a hibákat melyek nem voltak annyira lényegesek, hogy a tesztdokumentációba kerüljenek.

Itt értjük a keresési hibákat, azokat, ahol a fájl nem találta a végcélját.

Még értjük a logikai hibákat, ahol csak a kódban lévő kisebb elírások voltak.

Illetve azokat a hibákat is értjük itt, ahol a fájlhoz csatolt fájl, kép, netán link, nem volt található, mivel vagy rosszul importáltuk, vagy rossz mappába raktuk (képre gondolva), illetve hibás linket is megadtunk párszor, ahol már az oldal linkje, vagy domainje már hibás vagy elavult volt, így a lekérés hibásnak bizonyult.

Kis hibákról beszélünk itt és olyan ritkákról, amiket lényegtelen lenne le dokumentálni, mivel vagy nem tudtuk őket újra készíteni, vagy csak szimplán annyira sokszor tudtuk, hogy ez egy normális hibának minősült, aminek dokumentálásával nem jutottunk volna előre.

Beletartozhatnak ezek is a dokumentációba, csak akkor vagy 70 sor lenne ugyanabból az elemből, mivel ezek mindig más helyen jöttek elő, de a végkimenetel mindig ugyanaz maradt.

# **Összegzés**

Összegzés 2 pontra szednénk: Csapatmunka, feladatelvégzés.

Gyors talpaló: a csapat összhangban dolgozott és kevés vagy alig valamennyi hiba merült fel csak.

1. Csapatmunka

A csapatunk képes volt elosztani az adott feladatokat és megbírtuk beszélni azt, hogy végzi mindenki a saját kiadott dolgát, viszont egymásnak segítünk és bármi lenne akkor együtt is megcsinálunk egy-egy feladatot.

A kommunikálás sikerült nagyobb mennyiségben bár Gergőnek ez nem teljesen volt egy elérhető opció technikai eszköz hiánya miatt, így ő csak iskolában tudott dolgozni főképp, de ennek ellenére a csapat elég könnyen és kitartóan csinálta meg mind a WPF, JS feladatot, ami kiadott volt.

1. Feladatelvégzés

A projekt elvégzése először nehéz volt, mivel a feladatot nem teljesen értette elejében semmi, de kis idő múlva mikor a végcél realizálódott, így elbírtuk kezdeni a feladatelkészítését.

A WPF ment a leggördülékenyebben és ezt volt az a feladat, amivel rendkívül gyorsan tudtunk végezni, együtt csináltuk az elején aztán átadtuk Benedeknek a munkát, mivel ő vállalta az elvégzését, és probléma nélkül megbírta csinálni

A JavaScript/HTML oldal készítés kicsit a nehezebb síkon volt található, de pont ezért vállalta Andris, ugyanúgy itt is a nehézsége miatt csapatban dolgoztunk azán adtuk át Andrásnak a munkát, mikor már csak a könnyebben megoldható részek voltak már csak.

Összegezve a csapat kiválóan dolgozott és hibát mondani nem lehetne mind a dolgozási folyamattal kapcsolatban, és mind a kommunikáció és egymás megértésével kapcsolatban.

# **Felhasznált irodalom**

[The WPF Menu control - The complete WPF tutorial (wpf-tutorial.com)](https://wpf-tutorial.com/common-interface-controls/menu-control/)

[How To Create a Search Menu (w3schools.com)](https://www.w3schools.com/howto/howto_js_search_menu.asp)

[How To Build a Website Search Bar with JavaScript | Envato Tuts+ (tutsplus.com)](https://webdesign.tutsplus.com/how-to-build-a-search-bar-with-javascript--cms-107227t)

[Test Documentation in Software Testing (Example) (guru99.com)](https://www.guru99.com/testing-documentation.html)

[IT Documentation Software | IT Docs Tool | NinjaOne](https://www.ninjaone.com/it-documentation/?utm_source=bing&utm_medium=cpc&utm_campaign=EU_EN_PS_Feature_Documentation_RestOfEMEA&utm_content=demo&utm_term=IT%20documentation&utm_matchtype=p&utm_device=c&utm_adposition=&utm_gclid=&cq_src=bing_ads&cq_cmp=EU_EN_PS_Feature_IT%20Documentation_RestOfEMEA&cq_con=IT%20Documentation%20-%20Core&cq_term=IT%20documentation&cq_med=&cq_net=o&cq_plt=bp&_bt=&_bk=IT%20documentation&_bm=p&_bn=o&_bg=1139095000514578&msclkid=c76449d079551d9370cd2ff095dd6604)

[JavaScript Grid: Quick Start | AG Grid (ag-grid.com)](https://www.ag-grid.com/javascript-data-grid/getting-started/)

[WPF Menus A Complete Tutorial (c-sharpcorner.com)](https://www.c-sharpcorner.com/uploadfile/mahesh/menus-in-wpf/)

WPF. Jegyzet

[How to Write Test Cases: A Step-by-Step QA Guide | Coursera](https://www.coursera.org/articles/how-to-write-test-cases)

[CSS Grid Layout (w3schools.com)](https://www.w3schools.com/css/css_grid.asp)

[JavaScript Constructors (w3schools.com)](https://www.w3schools.com/JS/js_object_constructors.asp)

[c# - How do I show a Save As dialog in WPF? - Stack Overflow](https://stackoverflow.com/questions/5622854/how-do-i-show-a-save-as-dialog-in-wpf)

[Save content of a visual Object as a image file in WPF? - Stack Overflow](https://stackoverflow.com/questions/3617754/save-content-of-a-visual-object-as-a-image-file-in-wpf)