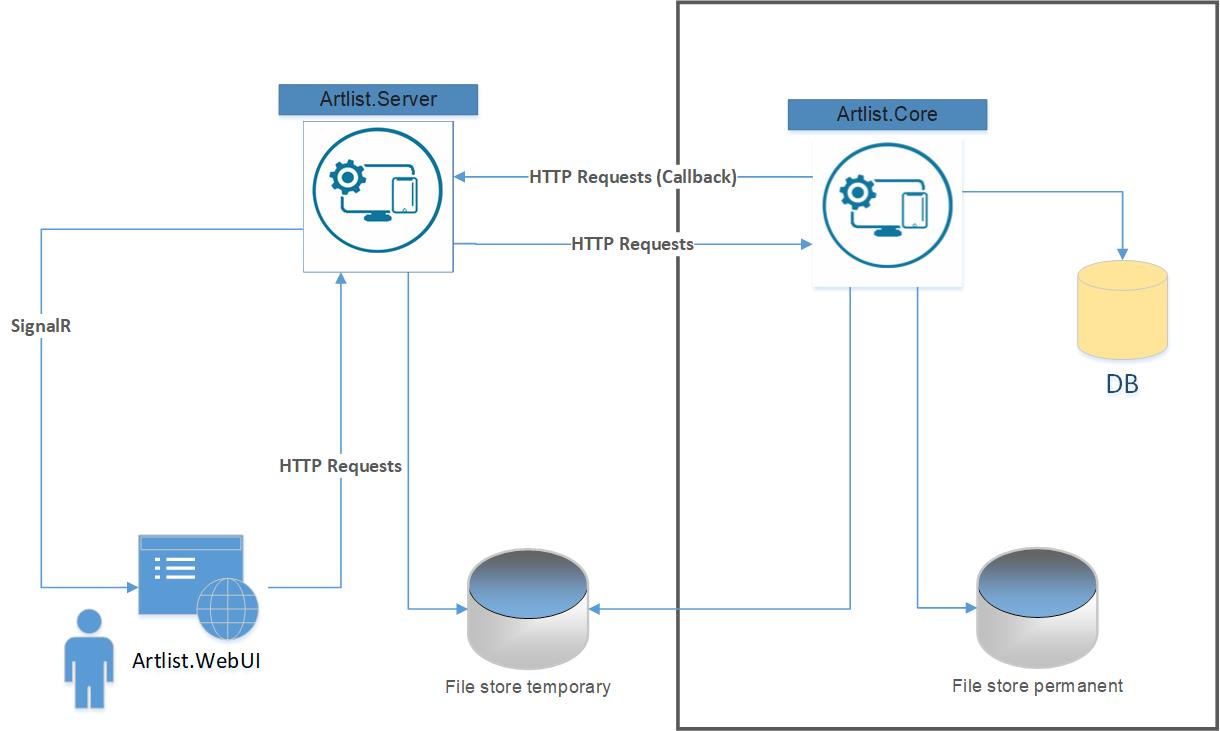
## התקנה והפעלה :

1. Artlist.WebUI
   1. תחת src\ environments יש להגדיר את ה URL של ה Artlist.Server
   2. להריץ “npm i” להתקנת כל ה packages
2. Artlist.Server
   1. בקובץ appsettings.json יש להגדיר את ה :
      1. CORS – כתובת ה Artlist.WebUI
      2. בקטגוריה ArtlisCoreServer את הכתובת של ה Artlist.Core וכתובת ל callback (כתובת של עצמו)
      3. בקטגוריה Files – להגדיר נתיב ל עבודה עם קבצים
3. Artlist.Core
   1. בקובץ appsettings.json יש להגדיר את ה :
      1. בקטגוריה Files – להגדיר נתיב ל עבודה עם קבצים
      2. בקטגוריה Database – להגדיר את ה connectionString
      3. בקטגוריה FFmpeg– תיקיה עם קבצי ה ffmpeg
4. להתקנת בסיס נתונים, השתמשתי ב SQL SERVER והוספתי סקריפט לבניית ה DB  
   כמו כן ליתר ביטחון הוספתי קובץ bak נקי

## חילוק ראשי

1. Frontend
   1. Artlist.WebUI
2. Backend
   1. Artlist.Core
   2. Artlist.Server
   3. Artlist.Common



הרצון שלי היה לדמות כמה שיותר מערכת production, לכן חילקתי את הפרויקט לא רק ל backend ו frontend אלא הוצאתי את כל ה BL לתהליך/פרויקט נוסף ה Artlist.Core -

Coreיכול לשבת מאחורי Firewall והוא גם היחיד שיש לו גישה לבסיס הנתונים.  
ה Artlist.Serverמשמש רק כ client או proxy בין ה webapp ל core  
בצורה זאת מאפשר לחבר בעתיד clients שונים עם דרישות שונות .  
Core גם היחיד בעל לוגיקת עיבוד הקבצים.

## תקשורת בין ה מודולים:

משתמש העובד דרך WebUI (SPA) פונה ל ה Server בפניות GET ו POST  
משם הבקשה עוברת מה Server ל Core באופן אסינכרוני.

הבקשה נכנסת לתור בקשות TaskEngine וכוללת URL לפניית callback חזרה (נניח שיש רשימת   
URL ש Core מכיר ומאשר)

אם יש צורך ל Core לעדכן, הוא פונה ל URL משם (במקרה זה ה Server) מעדכן את הלקוח  
ע"י תקשורת SignalR

## שמירת קבצים:

ישנם (תיאורטית) שני מקומות שונים לשמירת קבצים, האחד הוא ה temporary המדמה מקום זמני ופחות מאובטח והשני permanent שם יהיו רוב הקבצים (כרגע זאת אותה התיקייה והחילוק הוא רק בתתי התיקיות, אולם יש תשתית).  
ל server תהיה גישה רק ל temporary וזה לצורך שמירה מהירה על הקבצים המגיעים מהלקוח.  
Core יודע לגשת ל temporary וכך להתחיל לבצע עיבוד של המידע וכמובן גם ל permanent לשמירה ועיבוד.  
היה גם כדאי שביצוע ההמרה יעבוד עם ה temporary עד לסיום התהליך וזה כדאי שבמקרה של כישלון הקובץ הנוצר (חלקית) ימחק או אם הצליח, יעבור למקום קבוע.

ישנה אבסטרקציה IFileStore המאפשרת שמירה והעלאה של קבצים בעזרת מזהה GUID ייחודי  
מה שיאפשר לעבוד אם כך שרות שמירת קבצים בעתיד, כרגע המימוש הוא רק על גבי HDD.  
ניסיתי גם לעבוד עם stream בכדאי לעשות את הפעולות וזרימת המידע כמה שיותר גנרי.

# מספר הערות:

1. ישנה התנהגות מוזרה ל plugin העלאת הקבצים , כך שלא ניתן לבחור אותו קובץ פעמיים ברציפות
2. באופן אידיאלי היה כדאי למנוע עבודה על אותו קובץ בתהליכים נפרדים, או עבודה על קבצים שונים על HDD יחיד כיוון שזה פוגע ביעילות העבודה.
3. לא הספקתי לממש את כל פעולות ה repository (כרגע לא היה בהם צורך)
4. לא הספקתי למפות עד הסוף אץ ההקשרים בין האובייקטים ב Entity Framework
5. ברור לי שלא תמיד רצוי להשתמש בrepository אם עובדים עם Entity Framework אולם זה  
   מאפשר החלפת השיטה בעתיד.
6. הייתי מעדיף להחזיר רקviewmodels להשתמש ב automapper אולם לא הספקתי לעשות כן.
7. תהליך תפיסת הפריים בזמן ספציפי אינו מדייק תמיד.
8. כרגע כל ההודעות מגיעות לכל המשתמשים אולם ניתן לשמור את המזהה של הפונה ב server בתוספת של מזהה ה request וכך בתהליך ה calback להפנות את התשובה למשתמש ספציפי.
9. ישנה אפשרות לראות אחוזי התקדמות ולשלוח גם אותם למשתמש כך שיראה אותם בזמן אמת, אך טרם הספקתי.