IV Visualizer

Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB, Josua Benjamin Eyl, Lukas Friedrich, Max Bretschneider, Nathaniel Till Hartmann, Robin Köchel

Betreut von: Mickael Cormier M.Sc., Stefan Wolf M.Sc

IV Visualizer

Problemstellung Visualisierung und Verwaltung von Kamerastreams in Verbindung mit Video-Analyse KIs

Vorgehen und Lösungsansatz

- Model-View-Controller als Architekturmuster
  - Trennung der Verantwortlichkeiten
  - Flexibilität/Erweiterbarkeit
  - Leichte Wartbarkeit
  - Skalierbarkeit
- Verwendung von ScyllaDB
- gRPC als effizientes Kommunikationstool
- Änderungen am Pflichtenheft
  - Diagramme sind Kann-Kriterium geworden
  - Produktumgebung wurde angepasst
  - Projektverwaltung ist Kann-Kriterium

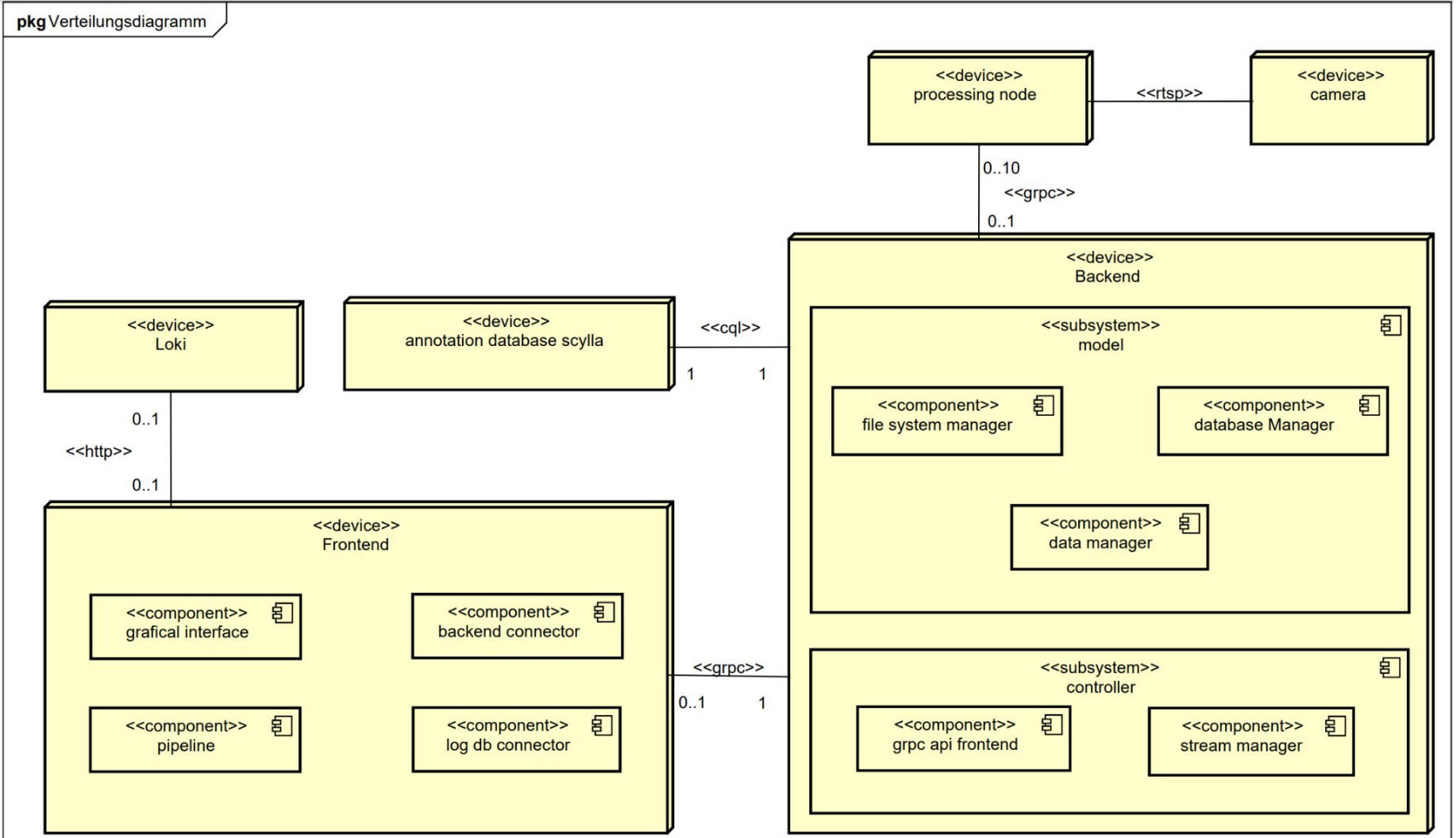
gRPC

- Zur Kommunikation zwischen Backend und Processing Node sowie Frontend und Backend verwendet
- Zwei unterschiedliche proto-Dateien zur Realisierung der Kommunikation
- proto-Dateien enthalten:
  - Die Definition des gRPC-Services
  - Die Nachrichten zwischen Client und Server sowie deren Attribute
  - Benötigte Enums

Architekturbeschreibung

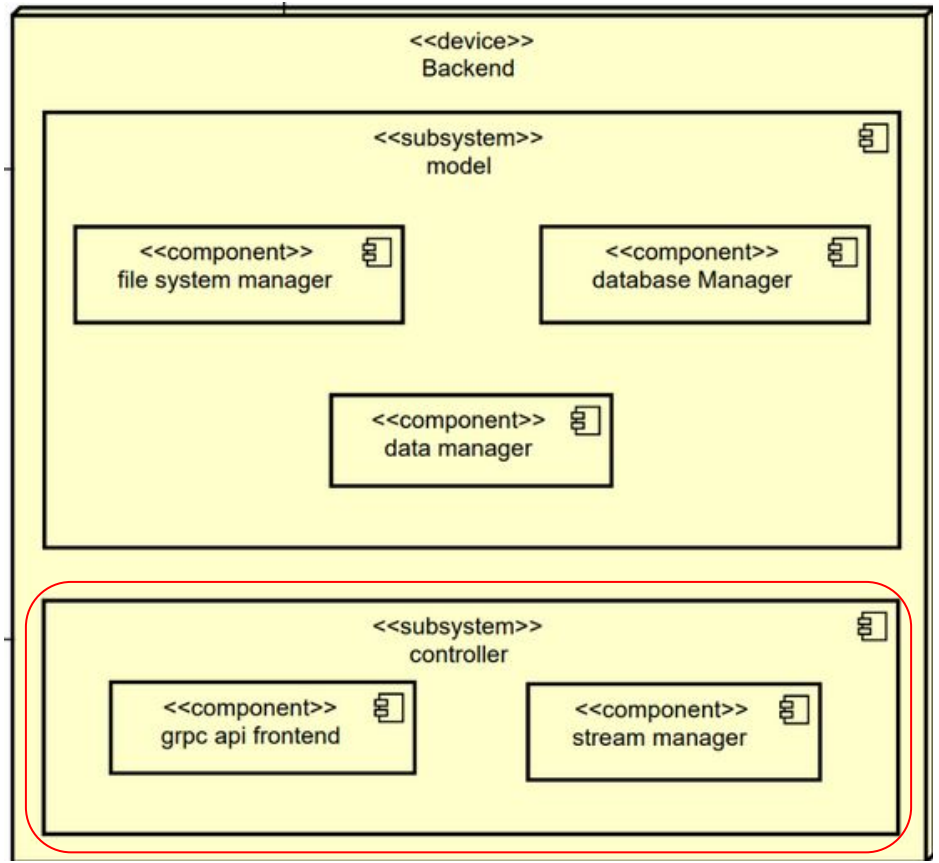
Model-View-Controller

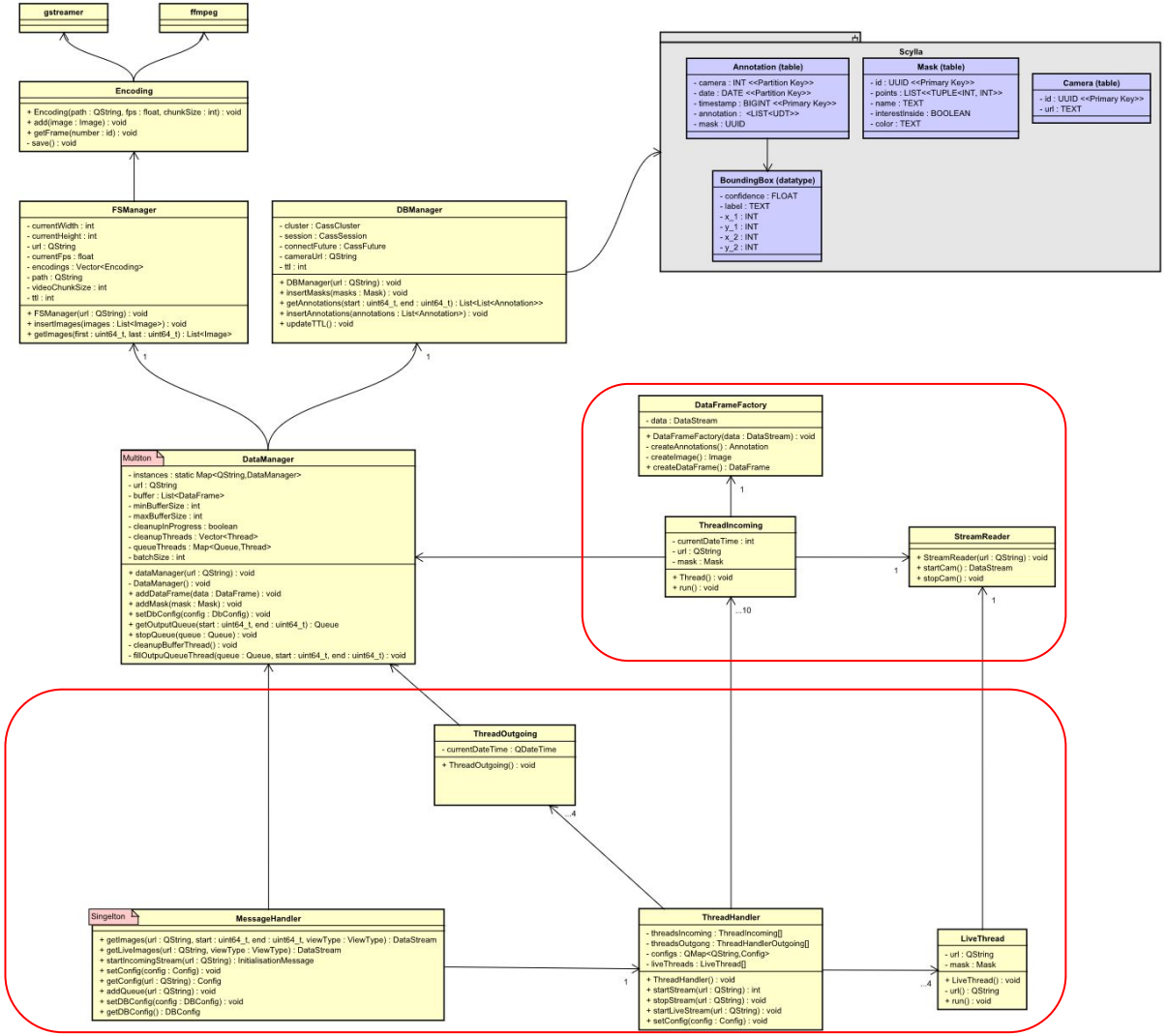
1. **Model:** Kümmert sich um das Speichern der Daten sowie um den Zugriff auf bereits abgelegte Daten
2. **View:** Zeigt die Daten an und nimmt Benutzereingaben entgegen, um diese an die Kontrolleinheit weiterzuleiten und verarbeiten zu lassen
3. **Controller:** Ist für das Entgegennehmen, Verarbeiten und Weiterleiten der Daten zuständig

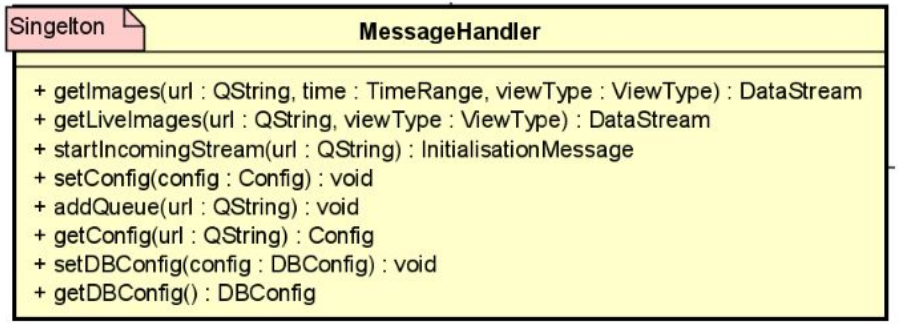

pkg Verteilungsdiagramm
<<device>>
processing node
<<rtsp>>
<<device>>
camera
0..10
<<grpc>>
0..1
<<device>>
Backend
<<device>>
Loki
<<device>>
annotation database scylla
<<cql>>
1
1
<<subsystem>>
model
<<component>>
file system manager
<<component>>
database Manager
<<component>>
data manager
0..1
<<http>>
0..1
<<device>>
Frontend
<<component>>
grafical interface
<<component>>
backend connector
<<component>>
pipeline
<<component>>
log db connector
<<grpc>>
0..1
1
<<subsystem>>
controller
<<component>>
grpc api frontend
<<component>>
stream manager

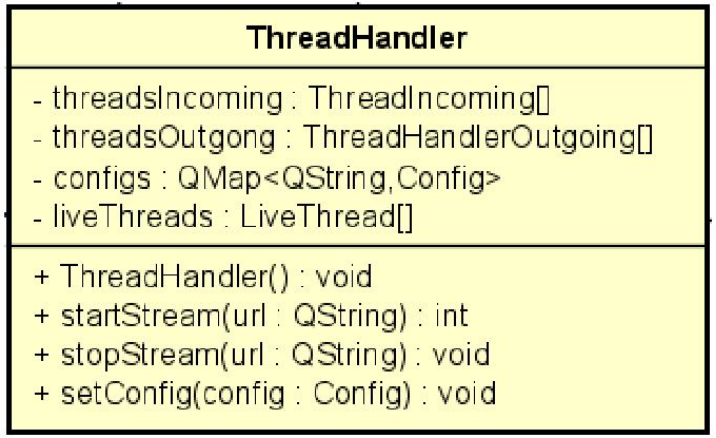
Controller

- Schnittstelle zu gRPC Verbindungen
- Kontrolliert Traffic von und in die Datenbank
- Reagiert auf Anfragen der GUI, beispielsweise zum Setzen der Config



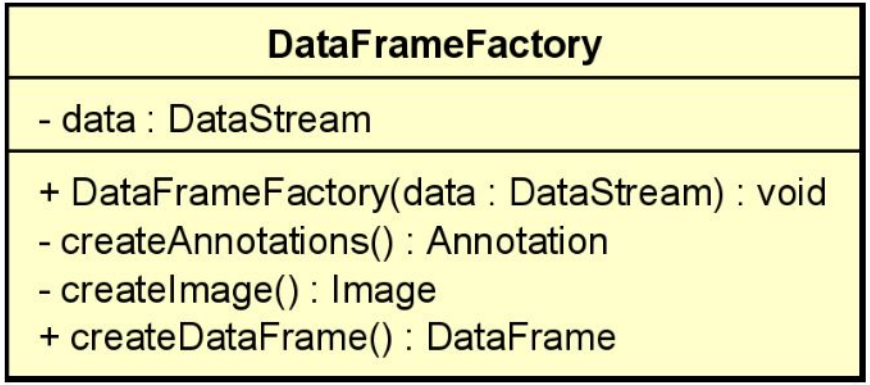

gstreamer
ffmpeg
Encoding
+ Encoding(path : QString, fps : float, chunkSize : int) : void
+ add(image : Image) : void
+ getFrame(number : id) : void
- save() : void
Scylla
Annotation (table)
- camera : INT <<Partition Key>>
- date : DATE <<Partition Key>>
- timestamp : BIGINT <<Primary Key>>
- annotation : <LIST<UDT>>
- mask : UUID
Mask (table)
- id : UUID <<Primary Key>>
- points : LIST<TUPLE<INT, INT>>
- name : TEXT
- interestInside : BOOLEAN
- color : TEXT
Camera (table)
- id : UUID <<Primary Key>>
- url : TEXT
BoundingBox (datatype)
- confidence : FLOAT
- label : TEXT
- x_1 : INT
- y_1 : INT
- x_2 : INT
- y_2 : INT
FSManager
- currentWidth : int
- currentHeight : int
- url : QString
- currentFps : float
- encodings : Vector<Encoding>
- path : QString
- videoChunkSize : int
- ttl : int
+ FSManager(url : QString) : void
+ insertImages(images : List<Image>) : void
+ getImages(first : uint64_t, last : uint64_t) : List<Image>
DBManager
- cluster : CassCluster
- session : CassSession
- connectFuture : CassFuture
- cameraUrl : QString
- ttl : int
+ DBManager(url : QString) : void
+ insertMasks(masks : Mask) : void
+ getAnnotations(start : uint64_t, end : uint64_t) : List<List<Annotation>>
+ insertAnnotations(annotations : List<Annotation>) : void
+ updateTTL() : void
1
1
Multiton
DataManager
- instances : static Map<QString,DataManager>
- url : QString
- buffer : List<DataFrame>
- minBufferSize : int
- maxBufferSize : int
- cleanupInProgress : boolean
- cleanupThreads : Vector<Thread>
- queueThreads : Map<Queue,Thread>
- batchSize : int
+ dataManager(url : QString) : void
- DataManager() : void
+ addDataFrame(data : DataFrame) : void
+ addMask(mask : Mask) : void
+ setDbConfig(config : DbConfig) : void
+ getOutputQueue(start : uint64_t, end : uint64_t) : Queue
+ stopQueue(queue : Queue) : void
- cleanupBufferThread() : void
- fillOutpuQueueThread(queue : Queue, start : uint64_t, end : uint64_t) : void
DataFrameFactory
- data : DataStream
+ DataFrameFactory(data : DataStream) : void
- createAnnotations() : Annotation
- createImage() : Image
+ createDataFrame() : DataFrame
1
ThreadIncoming
- currentDateTime : int
- url : QString
- mask : Mask
+ Thread() : void
+ run() : void
...10
StreamReader
+ StreamReader(url : QString) : void
+ startCam() : DataStream
+ stopCam() : void
1
1
ThreadOutgoing
- currentDateTime : QDateTime
+ ThreadOutgoing() : void
...4
Singelton
MessageHandler
+ getImages(url : QString, start : uint64_t, end : uint64_t, viewType : ViewType) : DataStream
+ getLiveImages(url : QString, viewType : ViewType) : DataStream
+ startIncomingStream(url : QString) : InitialisationMessage
+ setConfig(config : Config) : void
+ getConfig(url : QString) : Config
+ addQueue(url : QString) : void
+ setDBConfig(config : DBConfig) : void
+ getDBConfig() : DBConfig
ThreadHandler
- threadsIncoming : ThreadIncoming[]
- threadsOutgoing : ThreadHandlerOutgoing[]
- configs : QMap<QString,Config>
- liveThreads : LiveThread[]
+ ThreadHandler() : void
+ startStream(url : QString) : int
+ stopStream(url : QString) : void
+ startLiveStream(url : QString) : void
+ setConfig(config : Config) : void
1
...4
LiveThread
- url : QString
- mask : Mask
+ LiveThread() : void
- url() : QString
+ run() : void


Singelton
MessageHandler
+ getImages(url : QString, time : TimeRange, viewType : ViewType) : DataStream
+ getLiveImages(url : QString, viewType : ViewType) : DataStream
+ startIncomingStream(url : QString) : InitialisationMessage
+ setConfig(config : Config) : void
+ addQueue(url : QString) : void
+ getConfig(url : QString) : Config
+ setDBConfig(config : DBConfig) : void
+ getDBConfig() : DBConfig


ThreadHandler
- threadsIncoming : ThreadIncoming[]
- threadsOutgong : ThreadHandlerOutgoing[]
- configs : QMap<QString,Config>
- liveThreads : LiveThread[]
+ ThreadHandler() : void
+ startStream(url : QString) : int
+ stopStream(url : QString) : void
+ setConfig(config : Config) : void

1


gRPC


DataFrameFactory
- data : DataStream
+ DataFrameFactory(data : DataStream) : void
- createAnnotations() : Annotation
- createImage() : Image
+ createDataFrame() : DataFrame

Model

Persistente Speicherung und Verwaltung der objektorientierten Datentypen

Herausforderungen:

- Viele Datenpunkte
- Große Daten

Model

Persistente Speicherung und Verwaltung der objektorientierten Datentypen

Herausforderungen:

- Viele Datenpunkte
- Große Daten

{ 10 Streams x 30 FPS x 1500 Annotationen
= 450.000 Datenpunkte pro Sekunde

Model

Persistente Speicherung und Verwaltung der objektorientierten Datentypen

Herausforderungen:

- Viele Datenpunkte
- Große Daten

{ 10 Streams x 30 FPS x 1500 Annotationen
= 450.000 Datenpunkte pro Sekunde

Model

Persistente Speicherung und Verwaltung der objektorientierten Datentypen

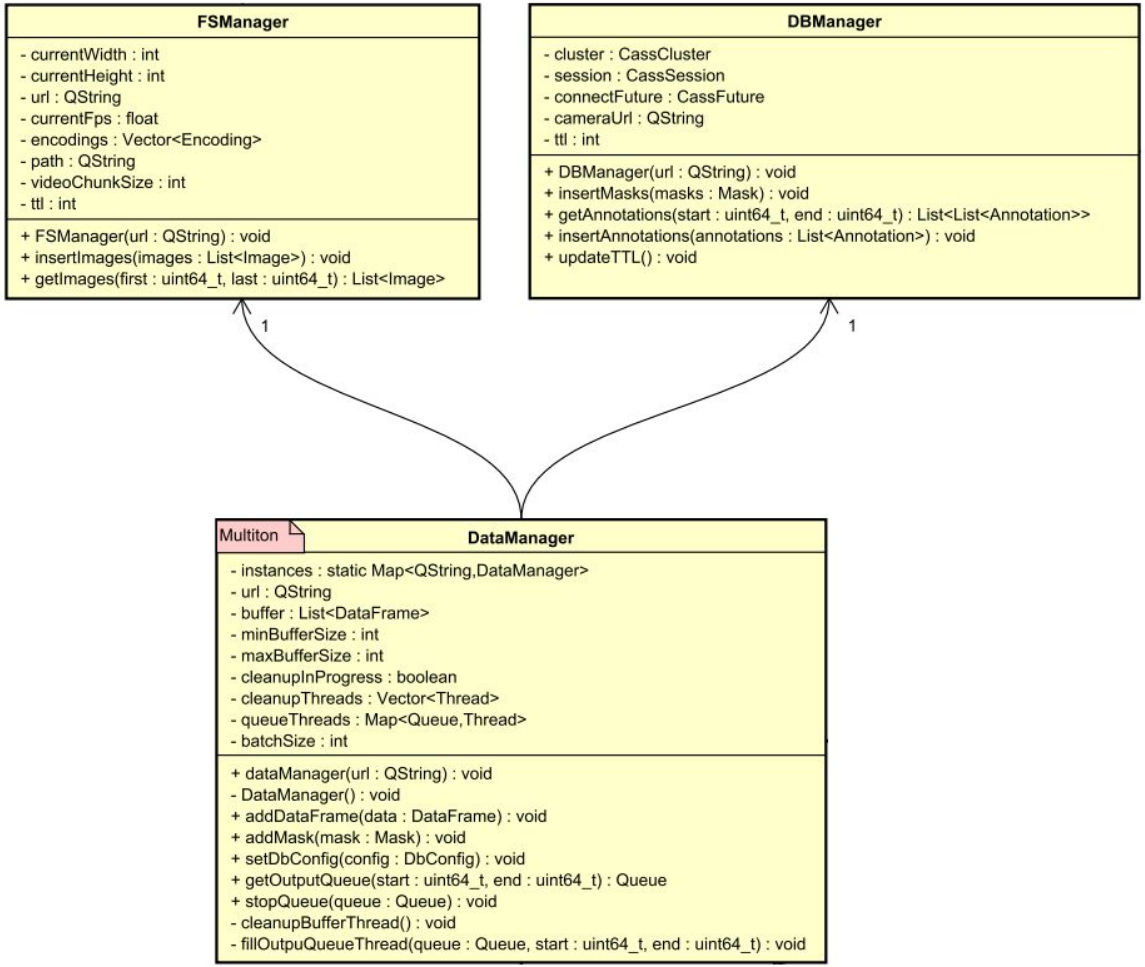
Herausforderungen:

- Viele Datenpunkte
- Große Daten

{ **Einzelbilder** der Auflösung 1920x1080
Bis zu 3,1 MB pro Bild
In der **Praxis** eher **400 kB** für komplizierte Bilder
⇒7,2 GB/Minute bei 10 Streams

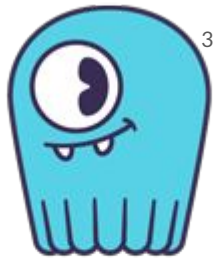
https://toolstud.io/photo/filesize.php?imagewidth=1920&imageheight=1080

Lösung


FSManager
- currentWidth : int
- currentHeight : int
- url : QString
- currentFps : float
- encodings : Vector<Encoding>
- path : QString
- videoChunkSize : int
- ttl : int
+ FSManager(url : QString) : void
+ insertImages(images : List<Image>) : void
+ getImages(first : uint64_t, last : uint64_t) : List<Image>
DBManager
- cluster : CassCluster
- session : CassSession
- connectFuture : CassFuture
- cameraUrl : QString
- ttl : int
+ DBManager(url : QString) : void
+ insertMasks(masks : Mask) : void
+ getAnnotations(start : uint64_t, end : uint64_t) : List<List<Annotation>>
+ insertAnnotations(annotations : List<Annotation>) : void
+ updateTTL() : void
1
1
Multiton
DataManager
- instances : static Map<QString,DataManager>
- url : QString
- buffer : List<DataFrame>
- minBufferSize : int
- maxBufferSize : int
- cleanupInProgress : boolean
- cleanupThreads : Vector<Thread>
- queueThreads : Map<Queue,Thread>
- batchSize : int
+ dataManager(url : QString) : void
- DataManager() : void
+ addDataFrame(data : DataFrame) : void
+ addMask(mask : Mask) : void
+ setDbConfig(config : DbConfig) : void
+ getOutputQueue(start : uint64_t, end : uint64_t) : Queue
+ stopQueue(queue : Queue) : void
- cleanupBufferThread() : void
- fillOutpuQueueThread(queue : Queue, start : uint64_t, end : uint64_t) : void

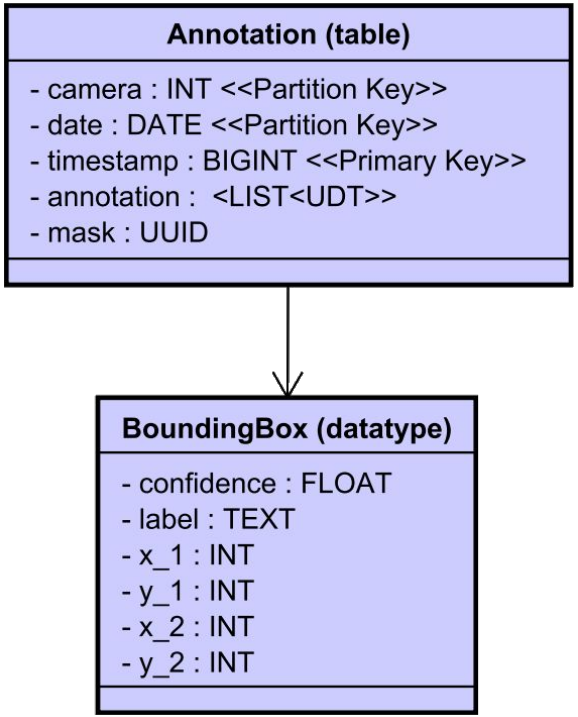
Scylla DB

1. Für Big Data optimiert
2. Horizontal skalierbar
3. Wide-column NoSQL Datenbank



SCYLLA

Scylla DB



SCYLLA

Speichern der Bilddaten

Umwandlung in Videos mit Enkodierung (Hardwarebeschleunigung)

Speichern der Bilddaten

Umwandlung in Videos mit Enkodierung (Hardwarebeschleunigung)

Video:	360 MB
Frames:	7250 MB

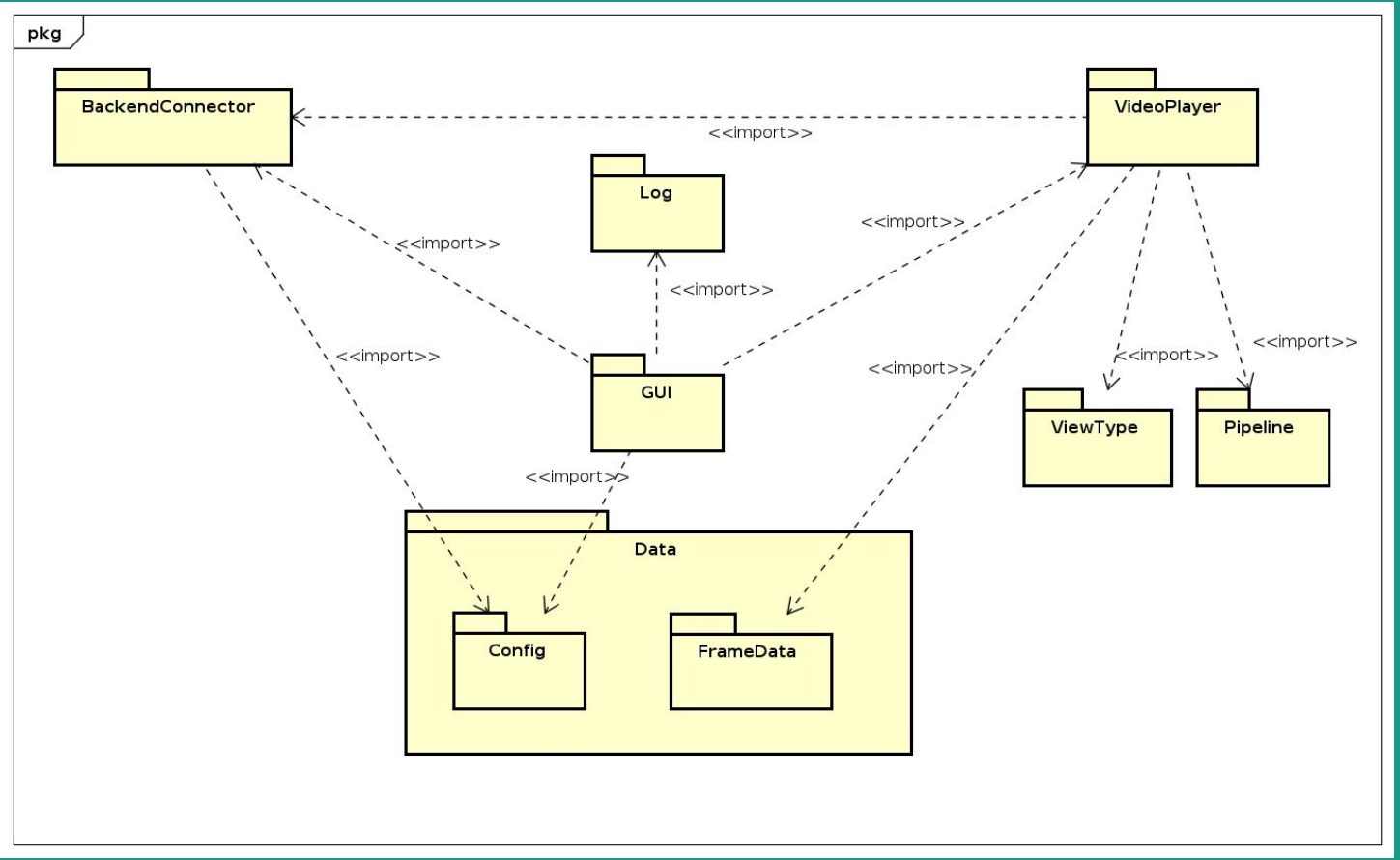
Speichern der Bilddaten

Umwandlung in Videos mit Enkodierung (Hardwarebeschleunigung)

Video:	360 MB
Frames:	7250 MB

Speicherung auf HDD

View


pkg
BackendConnector
VideoPlayer
<<import>>
Log
<<import>>
<<import>>
<<import>>
<<import>>
GUI
<<import>>
<<import>>
<<import>>
ViewType
Pipeline
<<import>>
Data
Config
FrameData

Log Datenbank mit Loki

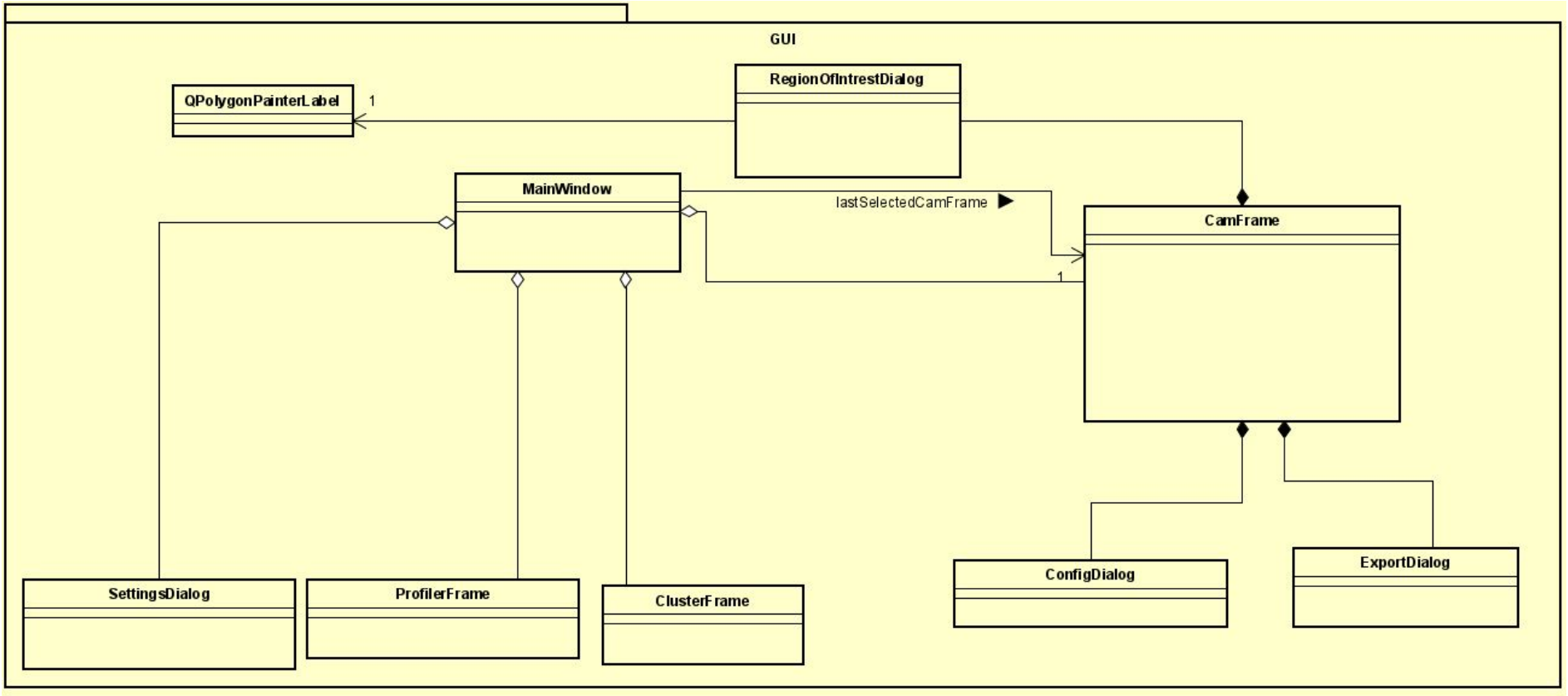
- Processing Nodes schreiben Logs in Datenbank
- Frontend stellt, abhängig von Filtern, Anfrage an Datenbank
- Filtermöglichkeiten:
  - Error
  - Warning
  - Debug
  - Detector
  - Preprocessor
  - Postprocessor

Dataframe

Jedes Objekt besteht aus:

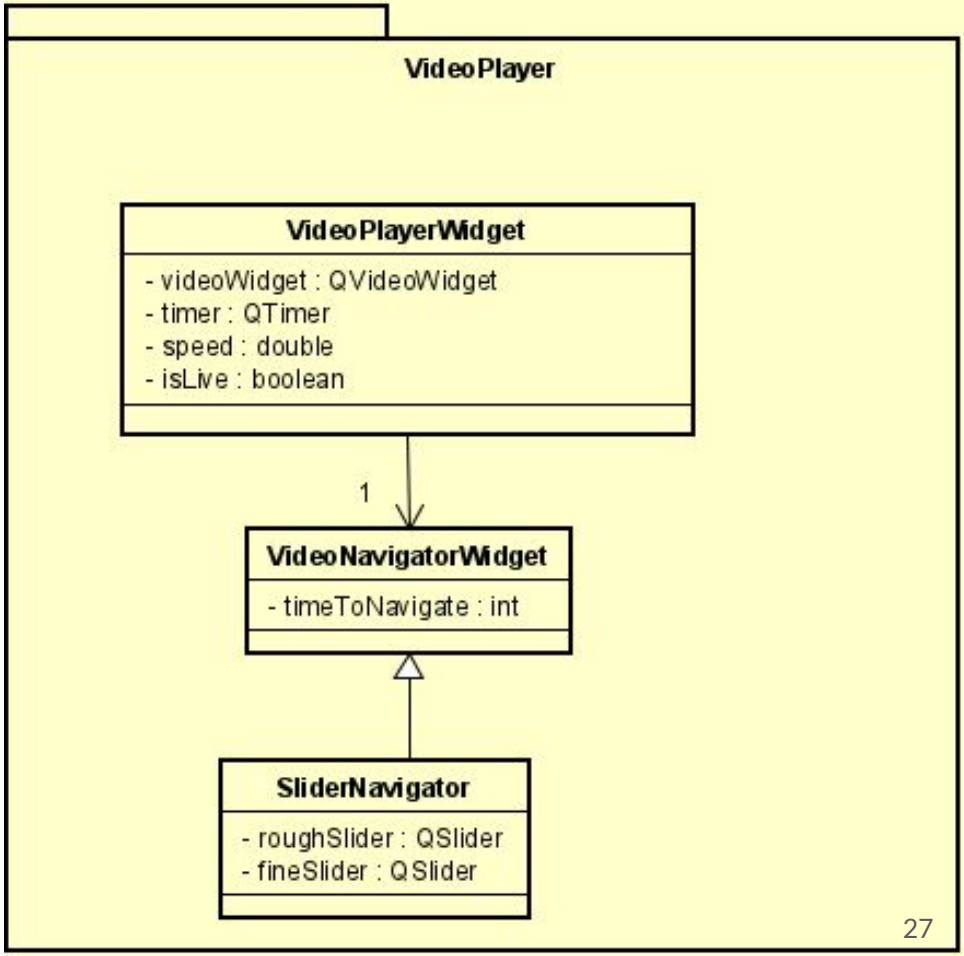
- Einer Kamera URL
- Einem Bild
- Menge von Annotationen
- Menge von ‘Region of Interest’

Pipeline


GUI
QPolygonPainterLabel
1
RegionOfIntrestDialog
MainWindow
lastSelectedCamFrame
CamFrame
1
SettingsDialog
ProfilerFrame
ClusterFrame
ConfigDialog
ExportDialog

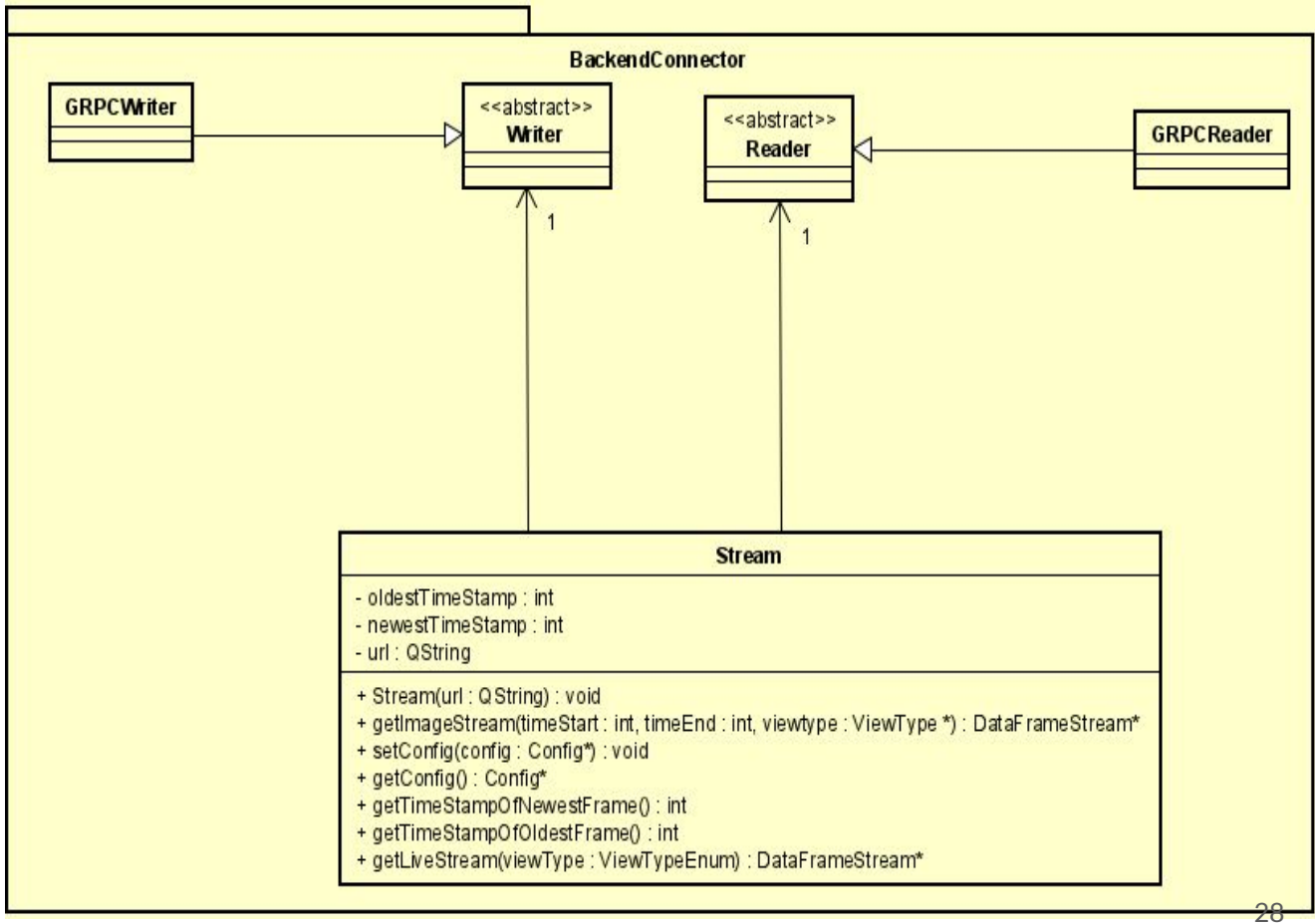
VideoPlayer

Zuständig für das Anzeigen des Streams



BackendConnector

Stellt anfragen an das **Backend**



Vielen Dank für die Aufmerksamkeit!

Bildquellen

1. https://grpc.io/
2. https://bilder.deutschlandfunk.de/FI/LE/_2/c2/FILE_2c26f748259f335178efab7eca80db8d/108594168-jpg-100-1920x1080.jpg
3. https://www.brighttalk.com/channel/18877/ (Scylla)
4. https://playgroundai.com/ Promt: “Factory hall with 3 workers and some machines. One on a ladder. Realistic drawing.”

Paralleler Zugriff

Implementierung der Klasse Encoding

- Cuts Strategie
- Buffering
- Verwendung von .fMP4 oder sogar .ts

(Frontend ist dafür ausgelegt)
((720 MB / Stream) / Minute)