DesktopApp

API

□ token: string

- login(username: string, password: string) : void
- signup(username: string, password: string, email: string) : void
- isAuthenticated() : boolean
- getAvailableTrainingParameters(): TrainingParameters
- train(options: IModelOptions) : boolean
- getTrainingProgress() : number
- getWorkspaces(): IWorkspace[]
- createWorkspace(w: string, s: ISensor[]): void
- getAvailableSensors(): ISensorSchema[]
- getWorkspaceSensors(): ISensor[]
- getSamples(w: IWorkspace): string[]
- deleteSample(w: IWorkspace, sample: string): void
- getDataCollectionURL(w: IWorkspace): string
- getPredictionURL(w: IWorkspace, m: string): string
- getLabels(w: IWorkspace): ILabel[]
- deleteWorkspace(workspace: IWorkspace): void
- createLabel(w: IWorkspace, l: string): void
- renameLabel(w: IWorkspace, l: ILabel, n: string): void
- describeLabel(w: IWorkspace, l: ILabel, d: string): void
- getSampleDetails(w: IWorkspace, sample: string): ISample
- setSampleLabel(w: IWorkspace, sample: string, label: string): void
- setSampleTimeframe(w: IWorkspace, sample: string, frame: Timeframe): void
- getModels(w: IWorkspace): string[]
- getModelMetrics(): IModelMetrics

«interfa...
IWorks...

id: strin...
name: s...

«component»
WorkspacePage

==props==
◇ workspace: string

«component»
WorkspaceModelDetailsPage

==props==
◇ workspace: string
◇ model: string

«component»
WorkspaceLabelsPage

==props==
◇ workspace: string

Workspa...

◇ worksp...
◇ model:...

«interface»
Workspace

: string,
ame: string

«interface»
ISensorSchema

name: string,
maxSamplingRate: number,
defaultSamplingRate: number,

«interface»
ILabel

id: string,
name: string,
description: string

«interface»
ISensorDatapoints

sensor

«component»
App

«component»
rkspaceModelClassifyPage

props
workspace: string
model: string

«component»
WorkspaceSampleDetailsPage

props
◇ workspace: string
◇ sample: string

«component»
SignupPage

state
◇ username: string
◇ password: string
◇ email: string

«com
WorkspaceC

p
◇ workspace

«interface»
ISensor

name: string,
samplingRate: number,

«interface»
IDatapoint

timestamp: Date
data: any

«interface»
IModelOptions

name: string,
features: string[],
imputation: string,
normalizer: string

«interface»
TrainingParameters

«interface»
ISample

label: ILabel,
data: ISensorDatapoints[]

«interface»
IModelMetrics

Timeframe

○ start: number
○ end: number

«component»
aceCollectDataPage

props
space: string

«component»
Header

props
◇ route: component

«component»
WorkspaceCreationPage

state
◇ selectedSensors: string[]
◇ samplingRates: int[]

«component»
WorkspaceModelsPag

props
◇ workspace: string

«interface»
Classifier

name: string
hyperparameters: [name: string, format: string]

*
classifier

...ne

...mber
...ber

«...t»
...lsPage

...ng

«component»
LoginPage

state

◇ username: string
◇ password: string

«component»
WorkspacesListPage

raviger

«component»
Link

«hook»
useRoutes

«component»
SampleList

====props====
◇ samples: ISample[]
◇ onCollect: (): void
◇ onDelete: (sample: ISample): void

«component»
ModelOptions

====props====
◇ onTrain: (options: IModelOptions): void
====state====
◇ options: IModelOptions

«component»
QRView

props
◇ text: string
◇ value: string

«component»
SampleChart

state
◇ frames: Timeframe[]
props
◇ sample: ISample
◇ onRelabel: (frames: Timeframe[]): void

react-qr-code

«component»
QRCode

highcharts

«component»
Link

«hook»
useRoutes