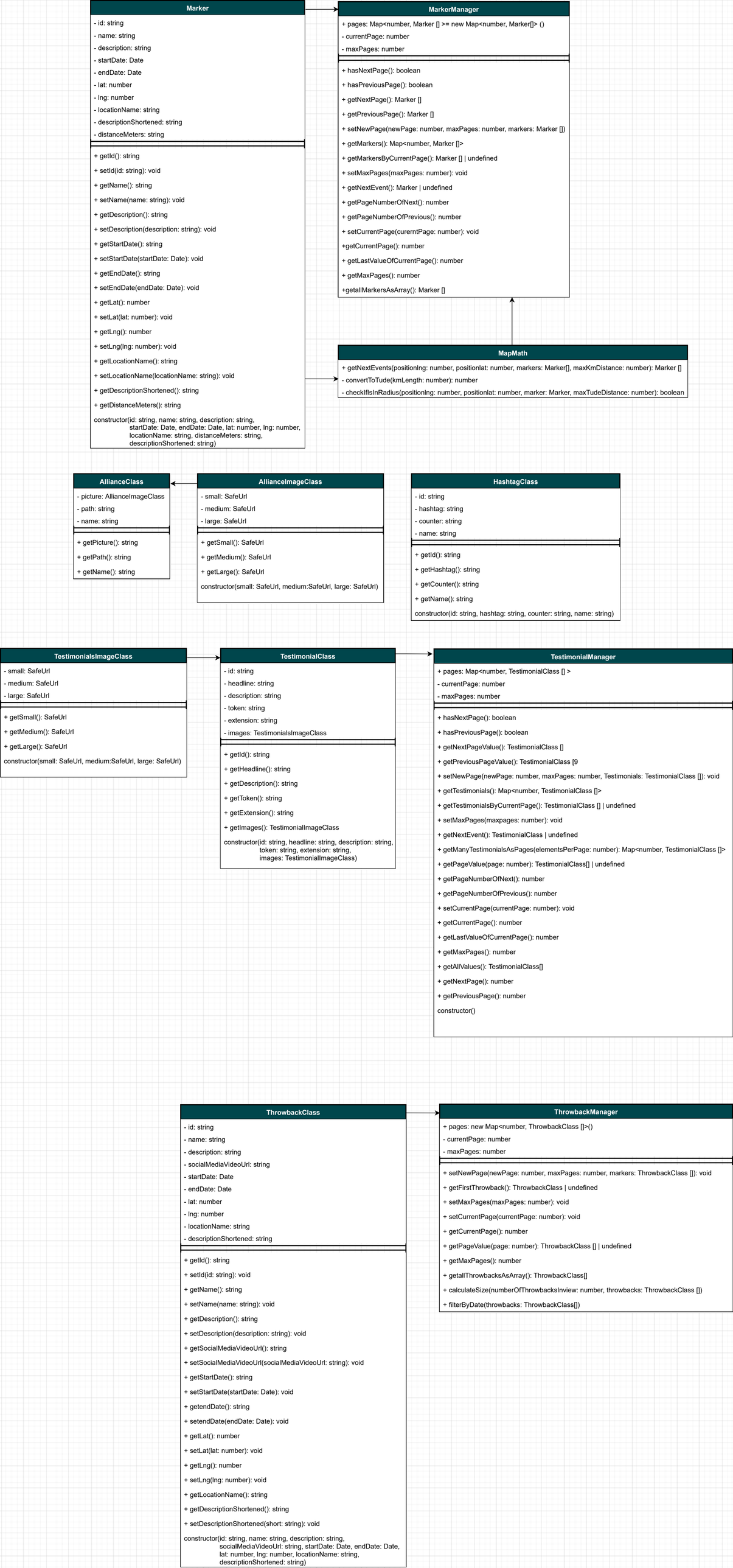
Marker
- id: string
- name: string
- description: string
- startDate: Date
- endDate: Date
- lat: number
- lng: number
- locationName: string
- descriptionShortened: string
- distanceMeters: string
+ getId(): string
+ setId(id: string): void
+ getName(): string
+ setName(name: string): void
+ getDescription(): string
+ setDescription(description: string): void
+ getStartDate(): string
+ setStartDate(startDate: Date): void
+ getEndDate(): string
+ setEndDate(endDate: Date): void
+ getLat(): number
+ setLat(lat: number): void
+ getLng(): number
+ setLng(lng: number): void
+ getLocationName(): string
+ setLocationName(locationName: string): void
+ getDescriptionShortened(): string
+ getDistanceMeters(): string
constructor(id: string, name: string, description: string,
startDate: Date, endDate: Date, lat: number, lng: number,
locationName: string, distanceMeters: string,
descriptionShortened: string)
MarkerManager
+ pages: Map<number, Marker [] >= new Map<number, Marker[]> ()
- currentPage: number
- maxPages: number
+ hasNextPage(): boolean
+ hasPreviousPage(): boolean
+ getNextPage(): Marker []
+ getPreviousPage(): Marker []
+ setNewPage(newPage: number, maxPages: number, markers: Marker [])
+ getMarkers(): Map<number, Marker []>
+ getMarkersByCurrentPage(): Marker [] | undefined
+ setMaxPages(maxPages: number): void
+ getNextEvent(): Marker | undefined
+ getPageNumberOfNext(): number
+ getPageNumberOfPrevious(): number
+ setCurrentPage(curerntPage: number): void
+getCurrentPage(): number
+ getLastValueOfCurrentPage(): number
+ getMaxPages(): number
+getallMarkersAsArray(): Marker []
MapMath
+ getNextEvents(positionlng: number, positionlat: number, markers: Marker[], maxKmDistance: number): Marker []
- convertToTude(kmLength: number): number
- checkIfIsInRadius(positionlng: number, positionlat: number, marker: Marker, maxTudeDistance: number): boolean
AllianceClass
- picture: AllianceImageClass
- path: string
- name: string
+ getPicture(): string
+ getPath(): string
+ getName(): string
AllianceImageClass
- small: SafeUrl
- medium: SafeUrl
- large: SafeUrl
+ getSmall(): SafeUrl
+ getMedium(): SafeUrl
+ getLarge(): SafeUrl
constructor(small: SafeUrl, medium:SafeUrl, large: SafeUrl)
HashtagClass
- id: string
- hashtag: string
- counter: string
- name: string
+ getId(): string
+ getHashtag(): string
+ getCounter(): string
+ getName(): string
constructor(id: string, hashtag: string, counter: string, name: string)
TestimonialsImageClass
- small: SafeUrl
- medium: SafeUrl
- large: SafeUrl
+ getSmall(): SafeUrl
+ getMedium(): SafeUrl
+ getLarge(): SafeUrl
constructor(small: SafeUrl, medium:SafeUrl, large: SafeUrl)
TestimonialClass
- id: string
- headline: string
- description: string
- token: string
- extension: string
- images: TestimonialsImageClass
+ getId(): string
+ getHeadline(): string
+ getDescription(): string
+ getToken(): string
+ getExtension(): string
+ getImages(): TestimonialImageClass
constructor(id: string, headline: string, description: string,
token: string, extension: string,
images: TestimonialImageClass)
TestimonialManager
+ pages: Map<number, TestimonialClass [] >
- currentPage: number
- maxPages: number
+ hasNextPage(): boolean
+ hasPreviousPage(): boolean
+ getNextPageValue(): TestimonialClass []
+ getPreviousPageValue(): TestimonialClass [9
+ setNewPage(newPage: number, maxPages: number, Testimonials: TestimonialClass []): void
+ getTestimonials(): Map<number, TestimonialClass []>
+ getTestimonialsByCurrentPage(): TestimonialClass [] | undefined
+ setMaxPages(maxpages: number): void
+ getNextEvent(): TestimonialClass | undefined
+ getManyTestimonialsAsPages(elementsPerPage: number): Map<number, TestimonialClass []>
+ getPageValue(page: number): TestimonialClass[] | undefined
+ getPageNumberOfNext(): number
+ getPageNumberOfPrevious(): number
+ setCurrentPage(currentPage: number): void
+ getCurrentPage(): number
+ getLastValueOfCurrentPage(): number
+ getMaxPages(): number
+ getAllValues(): TestimonialClass[]
+ getNextPage(): number
+ getPreviousPage(): number
constructor()
ThrowbackClass
- id: string
- name: string
- description: string
- socialMediaVideoUrl: string
- startDate: Date
- endDate: Date
- lat: number
- lng: number
- locationName: string
- descriptionShortened: string
+ getId(): string
+ setId(id: string): void
+ getName(): string
+ setName(name: string): void
+ getDescription(): string
+ setDescription(description: string): void
+ getSocialMediaVideoUrl(): string
+ setSocialMediaVideoUrl(socialMediaVideoUrl: string): void
+ getStartDate(): string
+ setStartDate(startDate: Date): void
+ getendDate(): string
+ setendDate(endDate: Date): void
+ getLat(): number
+ setLat(lat: number): void
+ getLng(): number
+ setLng(lng: number): void
+ getLocationName(): string
+ getDescriptionShortened(): string
+ setDescriptionShortened(short: string): void
constructor(id: string, name: string, description: string,
socialMediaVideoUrl: string, startDate: Date, endDate: Date,
lat: number, lng: number, locationName: string,
descriptionShortened: string)
ThrowbackManager
+ pages: new Map<number, ThrowbackClass []>()
- currentPage: number
- maxPages: number
+ setNewPage(newPage: number, maxPages: number, markers: ThrowbackClass []): void
+ getFirstThrowback(): ThrowbackClass | undefined
+ setMaxPages(maxPages: number): void
+ setCurrentPage(currentPage: number): void
+ getCurrentPage(): number
+ getPageValue(page: number): ThrowbackClass [] | undefined
+ getMaxPages(): number
+ getallThrowbacksAsArray(): ThrowbackClass[]
+ calculateSize(numberOfThrowbacksInview: number, throwbacks: ThrowbackClass [])
+ filterByDate(throwbacks: ThrowbackClass[])