

faketicket

Generated by Doxygen 1.9.3

<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>2</b>
2.1 Class Hierarchy	2
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 ticket Namespace Reference	9
5.1.1 Detailed Description	10
5.1.2 Typedef Documentation	10
5.1.3 Function Documentation	11
5.1.4 Variable Documentation	11
5.2 ticket::command Namespace Reference	13
5.2.1 Detailed Description	14
5.2.2 Typedef Documentation	14
5.2.3 Enumeration Type Documentation	14
5.2.4 Function Documentation	15
5.3 ticket::file Namespace Reference	17
5.3.1 Detailed Description	17
5.3.2 Variable Documentation	17
5.4 ticket::rollback Namespace Reference	18
5.4.1 Function Documentation	18
5.4.2 Variable Documentation	19
5.5 ticket::Station Namespace Reference	19
5.5.1 Typedef Documentation	19
<b>6 Class Documentation</b>	<b>19</b>
6.1 ticket::command::AddTrain Struct Reference	19
6.1.1 Member Data Documentation	20
6.2 ticket::rollback::AddTrain Struct Reference	21
6.2.1 Member Data Documentation	21
6.3 ticket::command::AddUser Struct Reference	21
6.3.1 Member Data Documentation	22
6.4 ticket::rollback::AddUser Struct Reference	22
6.4.1 Member Data Documentation	22
6.5 ticket::file::Array< T, maxLength, Cmp > Struct Template Reference	23
6.5.1 Detailed Description	24
6.5.2 Member Function Documentation	24

6.5.3 Member Data Documentation	26
6.6 ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk > Class Template Reference	26
6.6.1 Detailed Description	27
6.6.2 Constructor & Destructor Documentation	27
6.6.3 Member Function Documentation	27
6.7 ticket::command::BuyTicket Struct Reference	29
6.7.1 Member Data Documentation	29
6.8 ticket::rollback::BuyTicket Struct Reference	30
6.8.1 Member Data Documentation	30
6.9 ticket::command::Clean Struct Reference	31
6.10 ticket::Cmp< Lt > Class Template Reference	31
6.10.1 Detailed Description	31
6.10.2 Member Function Documentation	31
6.11 ticket::HashMap< Key, Value, Hash, Equal >::const_iterator Class Reference	33
6.11.1 Member Typedef Documentation	33
6.11.2 Constructor & Destructor Documentation	34
6.11.3 Member Function Documentation	34
6.11.4 Friends And Related Function Documentation	36
6.12 ticket::Vector< T >::const_iterator Class Reference	36
6.12.1 Member Typedef Documentation	37
6.12.2 Member Function Documentation	38
6.12.3 Friends And Related Function Documentation	40
6.13 ticket::Date Class Reference	40
6.13.1 Detailed Description	40
6.13.2 Constructor & Destructor Documentation	41
6.13.3 Member Function Documentation	41
6.14 ticket::Duration Class Reference	42
6.14.1 Detailed Description	43
6.14.2 Constructor & Destructor Documentation	43
6.14.3 Member Function Documentation	44
6.15 ticket::Train::Edge Struct Reference	45
6.15.1 Member Data Documentation	45
6.16 ticket::Exception Class Reference	45
6.16.1 Detailed Description	46
6.16.2 Constructor & Destructor Documentation	46
6.16.3 Member Function Documentation	46
6.17 ticket::command::Exit Struct Reference	46
6.18 ticket::file::File< Meta, szChunk > Class Template Reference	47
6.18.1 Detailed Description	47
6.18.2 Constructor & Destructor Documentation	47
6.18.3 Member Function Documentation	48

6.19 ticket::HashMap< Key, Value, Hash, Equal > Class Template Reference . . . . .	49
6.19.1 Detailed Description . . . . .	50
6.19.2 Member Typedef Documentation . . . . .	50
6.19.3 Constructor & Destructor Documentation . . . . .	51
6.19.4 Member Function Documentation . . . . .	51
6.20 ticket::file::Index< Key, Model, DataFile > Class Template Reference . . . . .	54
6.20.1 Detailed Description . . . . .	54
6.20.2 Constructor & Destructor Documentation . . . . .	54
6.20.3 Member Function Documentation . . . . .	55
6.21 ticket::file::Index< Varchar< maxLength >, Model, DataFile > Class Template Reference . . . . .	56
6.21.1 Detailed Description . . . . .	56
6.21.2 Constructor & Destructor Documentation . . . . .	56
6.21.3 Member Function Documentation . . . . .	57
6.22 ticket::Instant Class Reference . . . . .	58
6.22.1 Detailed Description . . . . .	58
6.22.2 Constructor & Destructor Documentation . . . . .	58
6.22.3 Member Function Documentation . . . . .	59
6.23 ticket::IOException Class Reference . . . . .	60
6.23.1 Constructor & Destructor Documentation . . . . .	60
6.24 ticket::HashMap< Key, Value, Hash, Equal >::iterator Class Reference . . . . .	60
6.24.1 Member Typedef Documentation . . . . .	61
6.24.2 Constructor & Destructor Documentation . . . . .	62
6.24.3 Member Function Documentation . . . . .	62
6.24.4 Friends And Related Function Documentation . . . . .	63
6.25 ticket::Vector< T >::iterator Class Reference . . . . .	64
6.25.1 Member Typedef Documentation . . . . .	65
6.25.2 Member Function Documentation . . . . .	65
6.25.3 Friends And Related Function Documentation . . . . .	67
6.26 ticket::rollback::LogEntry Struct Reference . . . . .	68
6.26.1 Member Data Documentation . . . . .	68
6.27 ticket::command::Login Struct Reference . . . . .	68
6.27.1 Member Data Documentation . . . . .	68
6.28 ticket::command::Logout Struct Reference . . . . .	69
6.28.1 Member Data Documentation . . . . .	69
6.29 ticket::file::ManagedObject< T, Meta, szChunk > Class Template Reference . . . . .	69
6.29.1 Detailed Description . . . . .	70
6.29.2 Constructor & Destructor Documentation . . . . .	70
6.29.3 Member Function Documentation . . . . .	70
6.30 ticket::Map< KeyType, ValueType, Compare > Class Template Reference . . . . .	71
6.30.1 Detailed Description . . . . .	72
6.30.2 Member Typedef Documentation . . . . .	72
6.30.3 Constructor & Destructor Documentation . . . . .	72

6.30.4 Member Function Documentation . . . . .	72
6.31 ticket::command::ModifyProfile Struct Reference . . . . .	75
6.31.1 Member Data Documentation . . . . .	75
6.32 ticket::rollback::ModifyProfile Struct Reference . . . . .	76
6.32.1 Member Data Documentation . . . . .	76
6.33 ticket::NotFound Class Reference . . . . .	76
6.33.1 Constructor & Destructor Documentation . . . . .	77
6.34 ticket::Optional< T > Class Template Reference . . . . .	77
6.34.1 Detailed Description . . . . .	78
6.34.2 Constructor & Destructor Documentation . . . . .	78
6.34.3 Member Function Documentation . . . . .	78
6.35 ticket::Order Struct Reference . . . . .	79
6.35.1 Member Typedef Documentation . . . . .	80
6.35.2 Member Enumeration Documentation . . . . .	80
6.35.3 Member Function Documentation . . . . .	80
6.35.4 Member Data Documentation . . . . .	80
6.36 ticket::OutOfBounds Class Reference . . . . .	81
6.36.1 Constructor & Destructor Documentation . . . . .	82
6.37 ticket::Overflow Class Reference . . . . .	82
6.37.1 Constructor & Destructor Documentation . . . . .	82
6.38 ticket::Pair< T1, T2 > Class Template Reference . . . . .	83
6.38.1 Detailed Description . . . . .	83
6.38.2 Constructor & Destructor Documentation . . . . .	83
6.38.3 Member Data Documentation . . . . .	84
6.39 ticket::ParseException Class Reference . . . . .	85
6.39.1 Constructor & Destructor Documentation . . . . .	85
6.40 ticket::PendingOrder Struct Reference . . . . .	85
6.40.1 Member Function Documentation . . . . .	86
6.40.2 Member Data Documentation . . . . .	86
6.41 ticket::command::QueryOrder Struct Reference . . . . .	87
6.41.1 Member Data Documentation . . . . .	87
6.42 ticket::command::QueryProfile Struct Reference . . . . .	87
6.42.1 Member Data Documentation . . . . .	87
6.43 ticket::command::QueryTicket Struct Reference . . . . .	88
6.43.1 Member Data Documentation . . . . .	88
6.44 ticket::command::QueryTrain Struct Reference . . . . .	88
6.44.1 Member Data Documentation . . . . .	89
6.45 ticket::command::QueryTransfer Struct Reference . . . . .	89
6.45.1 Member Data Documentation . . . . .	89
6.46 ticket::command::RefundTicket Struct Reference . . . . .	90
6.46.1 Member Data Documentation . . . . .	90
6.47 ticket::rollback::RefundTicket Struct Reference . . . . .	90

6.47.1 Member Data Documentation	90
6.48 ticket::command::ReleaseTrain Struct Reference	91
6.48.1 Member Data Documentation	91
6.49 ticket::rollback::ReleaseTrain Struct Reference	91
6.49.1 Member Data Documentation	91
6.50 ticket::Result< ResultType, ErrorType > Class Template Reference	92
6.50.1 Detailed Description	92
6.50.2 Constructor & Destructor Documentation	92
6.50.3 Member Function Documentation	93
6.51 ticket::Ride Struct Reference	93
6.51.1 Member Function Documentation	94
6.51.2 Member Data Documentation	94
6.52 ticket::RideSeats Struct Reference	94
6.52.1 Member Function Documentation	95
6.52.2 Member Data Documentation	95
6.53 ticket::command::Rollback Struct Reference	95
6.53.1 Member Data Documentation	96
6.54 ticket::file::Set< T, maxLength, Cmp > Struct Template Reference	96
6.54.1 Detailed Description	97
6.54.2 Constructor & Destructor Documentation	97
6.54.3 Member Function Documentation	97
6.54.4 Member Data Documentation	99
6.55 ticket::Train::Stop Struct Reference	99
6.55.1 Member Data Documentation	100
6.56 ticket::Train Struct Reference	100
6.56.1 Member Typedef Documentation	101
6.56.2 Member Function Documentation	101
6.56.3 Member Data Documentation	102
6.57 ticket::Underflow Class Reference	103
6.57.1 Constructor & Destructor Documentation	104
6.58 ticket::Unit Struct Reference	104
6.58.1 Detailed Description	104
6.58.2 Constructor & Destructor Documentation	104
6.58.3 Member Function Documentation	105
6.59 ticket::User Struct Reference	105
6.59.1 Member Typedef Documentation	106
6.59.2 Member Function Documentation	106
6.59.3 Member Data Documentation	106
6.60 ticket::file::Varchar< maxLength > Struct Template Reference	107
6.60.1 Detailed Description	108
6.60.2 Constructor & Destructor Documentation	108
6.60.3 Member Function Documentation	108

6.60.4 Friends And Related Function Documentation . . . . .	109
6.61 ticket::Variant< Ts > Class Template Reference . . . . .	110
6.61.1 Detailed Description . . . . .	111
6.61.2 Constructor & Destructor Documentation . . . . .	111
6.61.3 Member Function Documentation . . . . .	111
6.62 ticket::Vector< T > Class Template Reference . . . . .	113
6.62.1 Detailed Description . . . . .	114
6.62.2 Constructor & Destructor Documentation . . . . .	114
6.62.3 Member Function Documentation . . . . .	115
<b>7 File Documentation</b>	<b>118</b>
7.1 lib/algorithm.h File Reference . . . . .	118
7.1.1 Macro Definition Documentation . . . . .	118
7.2 algorithm.h . . . . .	119
7.3 lib/datetime.cpp File Reference . . . . .	119
7.4 lib/datetime.h File Reference . . . . .	119
7.5 datetime.h . . . . .	120
7.6 lib/exception.h File Reference . . . . .	120
7.7 exception.h . . . . .	121
7.8 lib/file/array.h File Reference . . . . .	122
7.9 array.h . . . . .	122
7.10 lib/file/bptree.h File Reference . . . . .	123
7.11 bptree.h . . . . .	124
7.12 lib/file/file.h File Reference . . . . .	130
7.13 file.h . . . . .	130
7.14 lib/file/index.h File Reference . . . . .	133
7.15 index.h . . . . .	133
7.16 lib/file/set.h File Reference . . . . .	134
7.17 set.h . . . . .	135
7.18 lib/file/varchar.h File Reference . . . . .	136
7.19 varchar.h . . . . .	136
7.20 lib/hashmap.h File Reference . . . . .	137
7.21 hashmap.h . . . . .	138
7.22 lib/map.h File Reference . . . . .	142
7.23 map.h . . . . .	142
7.24 lib/optional.h File Reference . . . . .	143
7.25 optional.h . . . . .	144
7.26 lib/result.h File Reference . . . . .	144
7.27 result.h . . . . .	145
7.28 lib/utility.cpp File Reference . . . . .	145
7.29 lib/utility.h File Reference . . . . .	145
7.29.1 Macro Definition Documentation . . . . .	146

7.30 utility.h . . . . .	147
7.31 lib/variant.h File Reference . . . . .	148
7.32 variant.h . . . . .	148
7.33 lib/vector.h File Reference . . . . .	150
7.34 vector.h . . . . .	150
7.35 src/main.cpp File Reference . . . . .	153
7.35.1 Function Documentation . . . . .	154
7.36 src/misc.cpp File Reference . . . . .	154
7.37 src/order.cpp File Reference . . . . .	154
7.38 src/order.h File Reference . . . . .	154
7.39 order.h . . . . .	155
7.40 src/parser.cpp File Reference . . . . .	155
7.41 src/parser.h File Reference . . . . .	156
7.42 parser.h . . . . .	157
7.43 src/rollback.cpp File Reference . . . . .	159
7.44 src/rollback.h File Reference . . . . .	159
7.45 rollback.h . . . . .	160
7.46 src/train.cpp File Reference . . . . .	161
7.47 src/train.h File Reference . . . . .	161
7.48 train.h . . . . .	162
7.49 src/user.cpp File Reference . . . . .	163
7.50 src/user.h File Reference . . . . .	163
7.51 user.h . . . . .	164
<b>Index</b>	<b>165</b>

## 1 Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>ticket</b>	<b>9</b>
<b>ticket::command</b>	
Classes and parsers for commands	<b>13</b>
<b>ticket::file</b>	
File utilities	<b>17</b>
<b>ticket::rollback</b>	<b>18</b>
<b>ticket::Station</b>	<b>19</b>



## 2 Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<code>ticket::command::AddTrain</code>	19
<code>ticket::rollback::AddTrain</code>	21
<code>ticket::command::AddUser</code>	21
<code>ticket::rollback::AddUser</code>	22
<code>ticket::file::Array&lt; T, maxLength, Cmp &gt;</code>	23
<code>ticket::file::Array&lt; int, 99 &gt;</code>	23
<code>ticket::file::Array&lt; Nodeld, 2 *k &gt;</code>	23
<code>ticket::file::Array&lt; ticket::Train::Edge, 99 &gt;</code>	23
<code>ticket::file::Array&lt; ticket::Train::Stop, 100 &gt;</code>	23
<code>ticket::file::BpTree&lt; KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk &gt;</code>	26
<code>ticket::file::BpTree&lt; Key, int &gt;</code>	26
<code>ticket::file::BpTree&lt; size_t, int &gt;</code>	26
<code>ticket::command::BuyTicket</code>	29
<code>ticket::rollback::BuyTicket</code>	30
<code>ticket::command::Clean</code>	31
<code>ticket::Cmp&lt; Lt &gt;</code>	31
<code>ticket::Cmp&lt;&gt;</code>	31
<code>ticket::HashMap&lt; Key, Value, Hash, Equal &gt;::const_iterator</code>	33
<code>ticket::Vector&lt; T &gt;::const_iterator</code>	36
<code>ticket::Date</code>	40
<code>ticket::Duration</code>	42
<code>ticket::Train::Edge</code>	45
<code>std::exception</code>	
<code>ticket::Exception</code>	45
<code>ticket::IoException</code>	60
<code>ticket::NotFound</code>	76
<code>ticket::OutOfBounds</code>	81
<code>ticket::Overflow</code>	82

ticket::Underflow	103
ticket::ParseException	85
ticket::command::Exit	46
ticket::file::File< Meta, szChunk >	47
ticket::file::File< Unit, kDefaultSzChunk >	47
ticket::HashMap< Key, Value, Hash, Equal >	49
ticket::HashMap< size_t, char * >	49
ticket::file::Index< Key, Model, DataFile >	54
ticket::file::Index< Varchar< maxLength >, Model, DataFile >	56
ticket::Instant	58
ticket::HashMap< Key, Value, Hash, Equal >::iterator	60
ticket::Vector< T >::iterator	64
ticket::command::Login	68
ticket::command::Logout	69
ticket::file::ManagedObject< T, Meta, szChunk >	69
ticket::file::ManagedObject< LogEntry >	69
ticket::rollback::LogEntry	68
ticket::file::ManagedObject< Node, Meta, szChunk >	69
ticket::file::ManagedObject< Order >	69
ticket::Order	79
ticket::file::ManagedObject< PendingOrder >	69
ticket::PendingOrder	85
ticket::file::ManagedObject< RideSeats >	69
ticket::RideSeats	94
ticket::file::ManagedObject< Train >	69
ticket::Train	100
ticket::file::ManagedObject< User >	69
ticket::User	105
ticket::Map< KeyType, ValueType, Compare >	71
ticket::command::ModifyProfile	75
ticket::rollback::ModifyProfile	76
ticket::Pair< T1, T2 >	83

ticket::Pair< const Key, Value >	83
ticket::command::QueryOrder	87
ticket::command::QueryProfile	87
ticket::command::QueryTicket	88
ticket::command::QueryTrain	88
ticket::command::QueryTransfer	89
ticket::command::RefundTicket	90
ticket::rollback::RefundTicket	90
ticket::command::ReleaseTrain	91
ticket::rollback::ReleaseTrain	91
ticket::Ride	93
ticket::command::Rollback	95
ticket::file::Set< T, maxLength, Cmp >	96
ticket::file::Set< Pair, 2 *k >	96
ticket::file::Set< Pair, 2 *l >	96
ticket::Train::Stop	99
ticket::Unit	104
ticket::file::Varchar< maxLength >	107
ticket::file::Varchar< 15 >	107
ticket::file::Varchar< 20 >	107
ticket::file::Varchar< 30 >	107
ticket::Variant< Ts >	110
ticket::Variant< ResultType, ErrorType >	110
ticket::Result< ResultType, ErrorType >	92
ticket::Variant< ticket::rollback::AddUser, ticket::rollback::ModifyProfile, ticket::rollback::AddTrain, ticket::rollback::ReleaseTrain, ticket::rollback::BuyTicket, ticket::rollback::RefundTicket >	110
ticket::Variant< Unit, int >	110
ticket::Optional< int >	77
ticket::Variant< Unit, std::string >	110
ticket::Optional< std::string >	77
ticket::Variant< Unit, T >	110
ticket::Optional< T >	77

<a href="#">ticket::Variant&lt; Unit, User::Privilege &gt;</a>	110
<a href="#">ticket::Optional&lt; User::Privilege &gt;</a>	77
<a href="#">ticket::Variant&lt; Unit, Varchar&lt; 15 &gt; &gt;</a>	110
<a href="#">ticket::Optional&lt; Varchar&lt; 15 &gt; &gt;</a>	77
<a href="#">ticket::Variant&lt; Unit, Varchar&lt; 30 &gt; &gt;</a>	110
<a href="#">ticket::Optional&lt; Varchar&lt; 30 &gt; &gt;</a>	77
<a href="#">ticket::Vector&lt; T &gt;</a>	113
<a href="#">ticket::Vector&lt; int &gt;</a>	113
<a href="#">ticket::Vector&lt; std::string &gt;</a>	113
<a href="#">ticket::Vector&lt; ticket::Date &gt;</a>	113
<a href="#">ticket::Vector&lt; ticket::Duration &gt;</a>	113

## 3 Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">ticket::command::AddTrain</a>	19
<a href="#">ticket::rollback::AddTrain</a>	21
<a href="#">ticket::command::AddUser</a>	21
<a href="#">ticket::rollback::AddUser</a>	22
<a href="#">ticket::file::Array&lt; T, maxLength, Cmp &gt;</a>	
An on-stack array with utility functions and bound checks	23
<a href="#">ticket::file::BpTree&lt; KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk &gt;</a>	
Implementation of the B+ tree	26
<a href="#">ticket::command::BuyTicket</a>	29
<a href="#">ticket::rollback::BuyTicket</a>	30
<a href="#">ticket::command::Clean</a>	31
<a href="#">ticket::Cmp&lt; Lt &gt;</a>	
Comparison utilities	31
<a href="#">ticket::HashMap&lt; Key, Value, Hash, Equal &gt;::const_iterator</a>	33
<a href="#">ticket::Vector&lt; T &gt;::const_iterator</a>	36
<a href="#">ticket::Date</a>	
Class representing a date between 2021-06-01 and 2021-08-31 (inclusive)	40

<a href="#">ticket::Duration</a>	42
Class representing a length of timespan	
<a href="#">ticket::Train::Edge</a>	45
<a href="#">ticket::Exception</a>	45
The base exception class	
<a href="#">ticket::command::Exit</a>	46
<a href="#">ticket::file::File&lt; Meta, szChunk &gt;</a>	47
A chunked file storage with manual garbage collection	
<a href="#">ticket::HashMap&lt; Key, Value, Hash, Equal &gt;</a>	49
An unordered hash-based map	
<a href="#">ticket::file::Index&lt; Key, Model, DataFile &gt;</a>	54
Class representing an index file	
<a href="#">ticket::file::Index&lt; Varchar&lt; maxLength &gt;, Model, DataFile &gt;</a>	56
Specialization of <a href="#">Index</a> on <a href="#">Varchar</a>	
<a href="#">ticket::Instant</a>	58
Class representing a point of time in a day	
<a href="#">ticket::IOException</a>	60
<a href="#">ticket::HashMap&lt; Key, Value, Hash, Equal &gt;::iterator</a>	60
<a href="#">ticket::Vector&lt; T &gt;::iterator</a>	64
<a href="#">ticket::rollback::LogEntry</a>	68
<a href="#">ticket::command::Login</a>	68
<a href="#">ticket::command::Logout</a>	69
<a href="#">ticket::file::ManagedObject&lt; T, Meta, szChunk &gt;</a>	69
Opinionated utility base class for the objects to be stored	
<a href="#">ticket::Map&lt; KeyType, ValueType, Compare &gt;</a>	71
A sorted key-value map backed by a red-black tree	
<a href="#">ticket::command::ModifyProfile</a>	75
<a href="#">ticket::rollback::ModifyProfile</a>	76
<a href="#">ticket::NotFound</a>	76
<a href="#">ticket::Optional&lt; T &gt;</a>	77
A resemblance of <code>std::optional</code>	
<a href="#">ticket::Order</a>	79
<a href="#">ticket::OutOfBounds</a>	81
<a href="#">ticket::Overflow</a>	82
<a href="#">ticket::Pair&lt; T1, T2 &gt;</a>	83
A pair of objects	
<a href="#">ticket::ParseException</a>	85

<a href="#">ticket::PendingOrder</a>	85
<a href="#">ticket::command::QueryOrder</a>	87
<a href="#">ticket::command::QueryProfile</a>	87
<a href="#">ticket::command::QueryTicket</a>	88
<a href="#">ticket::command::QueryTrain</a>	88
<a href="#">ticket::command::QueryTransfer</a>	89
<a href="#">ticket::command::RefundTicket</a>	90
<a href="#">ticket::rollback::RefundTicket</a>	90
<a href="#">ticket::command::ReleaseTrain</a>	91
<a href="#">ticket::rollback::ReleaseTrain</a>	91
<a href="#">ticket::Result&lt; ResultType, ErrorType &gt;</a> Result<Res, Err> = Res   Err	92
<a href="#">ticket::Ride</a>	93
<a href="#">ticket::RideSeats</a>	94
<a href="#">ticket::command::Rollback</a>	95
<a href="#">ticket::file::Set&lt; T, maxLength, Cmp &gt;</a> A sorted array with utility functions and bound checks	96
<a href="#">ticket::Train::Stop</a>	99
<a href="#">ticket::Train</a>	100
<a href="#">ticket::Underflow</a>	103
<a href="#">ticket::Unit</a> An empty class, used at various places	104
<a href="#">ticket::User</a>	105
<a href="#">ticket::file::Varchar&lt; maxLength &gt;</a> A wrapper for const char * with utility functions and type conversions	107
<a href="#">ticket::Variant&lt; Ts &gt;</a> A tagged union, aka sum type	110
<a href="#">ticket::Vector&lt; T &gt;</a> A data container like std::vector	113

## 4 File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">lib/algorithm.h</a>	118
---------------------------------	-----

---

<a href="#">lib/datetime.cpp</a>	119
<a href="#">lib/datetime.h</a>	119
<a href="#">lib/exception.h</a>	120
<a href="#">lib/hashmap.h</a>	137
<a href="#">lib/map.h</a>	142
<a href="#">lib/optional.h</a>	143
<a href="#">lib/result.h</a>	144
<a href="#">lib/utility.cpp</a>	145
<a href="#">lib/utility.h</a>	145
<a href="#">lib/variant.h</a>	148
<a href="#">lib/vector.h</a>	150
<a href="#">lib/file/array.h</a>	122
<a href="#">lib/file/bptree.h</a>	123
<a href="#">lib/file/file.h</a>	130
<a href="#">lib/file/index.h</a>	133
<a href="#">lib/file/set.h</a>	134
<a href="#">lib/file/varchar.h</a>	136
<a href="#">src/main.cpp</a>	153
<a href="#">src/misc.cpp</a>	154
<a href="#">src/order.cpp</a>	154
<a href="#">src/order.h</a>	154
<a href="#">src/parser.cpp</a>	155
<a href="#">src/parser.h</a>	156
<a href="#">src/rollback.cpp</a>	159
<a href="#">src/rollback.h</a>	159
<a href="#">src/train.cpp</a>	161
<a href="#">src/train.h</a>	161
<a href="#">src/user.cpp</a>	163
<a href="#">src/user.h</a>	163

## 5 Namespace Documentation

### 5.1 ticket Namespace Reference

#### Namespaces

- namespace [command](#)  
*Classes and parsers for commands.*
- namespace [file](#)  
*File utilities.*
- namespace [rollback](#)
- namespace [Station](#)

#### Classes

- class [Cmp](#)  
*Comparison utilities.*
- class [Date](#)  
*Class representing a date between 2021-06-01 and 2021-08-31 (inclusive).*
- class [Duration](#)  
*Class representing a length of timespan.*
- class [Exception](#)  
*The base exception class.*
- class [HashMap](#)  
*An unordered hash-based map.*
- class [Instant](#)  
*Class representing a point of time in a day.*
- class [IOException](#)
- class [Map](#)  
*A sorted key-value map backed by a red-black tree.*
- class [NotFound](#)
- class [Optional](#)  
*A resemblance of `std::optional`.*
- struct [Order](#)
- class [OutOfBounds](#)
- class [Overflow](#)
- class [Pair](#)  
*A pair of objects.*
- class [ParseException](#)
- struct [PendingOrder](#)
- class [Result](#)  
 *$\text{Result}<\text{Res}, \text{Err}> = \text{Res} \mid \text{Err}$ .*
- struct [Ride](#)
- struct [RideSeats](#)
- struct [Train](#)
- class [Underflow](#)
- struct [Unit](#)  
*An empty class, used at various places.*
- struct [User](#)
- class [Variant](#)  
*A tagged union, aka sum type.*
- class [Vector](#)  
*A data container like `std::vector`.*



## Typedefs

- `template<typename Lt = internal::LessOp>`  
`using Less = Cmp< Lt >`
- `template<typename Lt = internal::LessOp>`  
`using Greater = Cmp< internal::GreaterOp< Lt > >`

## Functions

- `auto split (std::string &str, char sep) -> Vector< std::string_view >`  
*splits the string with sep into several substrings.*
- `auto copyStrings (const Vector< std::string_view > &vec) -> Vector< std::string >`  
*copies the strings in vec into an array of real strings.*
- `template<typename T >`  
`auto declval () -> T`  
*declare value, used in type annotations.*
- `template<typename T >`  
`auto move (T &val) -> T &&`  
*forcefully make an rvalue.*

## Variables

- `file::File orders {"orders"}`
- `file::Index< User::Id, Order, decltype(orders)> ixOrdersUserId {&Order::user, "orders.user.ix", orders}`
- `file::File pendingOrders {"pending-orders"}`
- `file::Index< Ride, PendingOrder, decltype(pendingOrders)> ixPendingOrdersRide`
- `file::File logEntries {"rollback-log"}`
- `file::File trains {"trains"}`
- `file::Index< Train::Id, Train, decltype(trains)> ixTrainsId {&Train::trainId, "trains.train-id.ix", trains}`
- `file::BpTree< size_t, int > ixTrainsStop {"trains.stop.ix"}`
- `file::File rideSeats {"ride-seats"}`
- `file::Index< Ride, RideSeats, decltype(rideSeats)> ixRideSeatsRide`
- `file::File users {"users"}`
- `file::Index< User::Id, User, decltype(users)> ixUsersUsername {&User::username, "users.username.ix", users}`
- `HashMap< std::string, Unit > usersLoggedIn`  
*a set of users that are logged in.*
- `constexpr Unit unit`

### 5.1.1 Detailed Description

This file defines exception classes used throughout the project. Throwing exceptions is not encouraged, since it has a poor stack unwinding performance.

### 5.1.2 Typedef Documentation

**5.1.2.1 Greater** `template<typename Lt = internal::LessOp>`  
`using ticket::Greater = typedef Cmp<internal::GreaterOp<Lt> >`

**5.1.2.2 Less** `template<typename Lt = internal::LessOp>`  
`using ticket::Less = typedef Cmp<Lt>`

### 5.1.3 Function Documentation

**5.1.3.1 copyStrings()** `auto ticket::copyStrings (`  
`const Vector< std::string_view > & vec ) -> Vector< std::string >`

copies the strings in vec into an array of real strings.

**5.1.3.2 declval()** `template<typename T >`  
`auto ticket::declval ( ) -> T`

declare value, used in type annotations.

**5.1.3.3 move()** `template<typename T >`  
`auto ticket::move (`  
`T & val ) -> T &&`

forcefully make an rvalue.

**5.1.3.4 split()** `auto ticket::split (`  
`std::string & str,`  
`char sep ) -> Vector< std::string_view >`

splits the string with sep into several substrings.

this function mutates the incoming string to make sure the result is properly zero-terminated.

the lifetime of the return value is the lifetime of the incoming string; that is to say, you need to keep the original string from destructured in order to use the result.

### 5.1.4 Variable Documentation

**5.1.4.1 ixOrdersUserId** `file::Index< User::Id, Order, decltype(orders)> ticket::ixOrdersUserId {&Order::user, "orders.user.ix", orders}`

**5.1.4.2 ixPendingOrdersRide** `file::Index< Ride, PendingOrder, decltype(pendingOrders)> ticket::ixPendingOrdersRide`

**Initial value:**

```
{
    &PendingOrder::ride,
    "pending-orders.ride.ix",
    pendingOrders
}
```

**5.1.4.3 ixRideSeatsRide** `file::Index< Ride, RideSeats, decltype(rideSeats)> ticket::ixRideSeatsRide`

**Initial value:**

```
{
    &RideSeats::ride,
    "ride-seats.ride.ix",
    rideSeats
}
```

**5.1.4.4 ixTrainsId** `file::Index< Train::Id, Train, decltype(trains)> ticket::ixTrainsId {&Train::trainId, "trains.train-id.ix", trains}`

**5.1.4.5 ixTrainsStop** `file::BpTree< size_t, int > ticket::ixTrainsStop {"trains.stop.ix"}`

**5.1.4.6 ixUsersUsername** `file::Index< User::Id, User, decltype(users)> ticket::ixUsersUsername {&User::username, "users.username.ix", users}`

**5.1.4.7 logEntries** `file::File ticket::logEntries {"rollback-log"}`

**5.1.4.8 orders** `file::File ticket::orders {"orders"}`

**5.1.4.9 pendingOrders** `file::File` `ticket::pendingOrders {"pending-orders"}`

**5.1.4.10 rideSeats** `file::File` `ticket::rideSeats {"ride-seats"}`

**5.1.4.11 trains** `file::File` `ticket::trains {"trains"}`

**5.1.4.12 unit** `constexpr Unit` `ticket::unit [inline], [constexpr]`

**5.1.4.13 users** `file::File` `ticket::users {"users"}`

**5.1.4.14 usersLoggedIn** `HashMap<std::string, Unit>` `ticket::usersLoggedIn`

a set of users that are logged in.

## 5.2 ticket::command Namespace Reference

Classes and parsers for commands.

### Classes

- struct [AddTrain](#)
- struct [AddUser](#)
- struct [BuyTicket](#)
- struct [Clean](#)
- struct [Exit](#)
- struct [Login](#)
- struct [Logout](#)
- struct [ModifyProfile](#)
- struct [QueryOrder](#)
- struct [QueryProfile](#)
- struct [QueryTicket](#)
- struct [QueryTrain](#)
- struct [QueryTransfer](#)
- struct [RefundTicket](#)
- struct [ReleaseTrain](#)
- struct [Rollback](#)

## Typedefs

- using `Command` = `Variant< AddUser, Login, Logout, QueryProfile, ModifyProfile, AddTrain, ReleaseTrain, QueryTrain, QueryTicket, QueryTransfer, BuyTicket, QueryOrder, RefundTicket, Rollback, Clean, Exit >`

## Enumerations

- enum `SortType` { `kTime` , `kCost` }

## Functions

- auto `parse` (`std::string &str`) -> `Result< Command, ParseException >`  
*parses the command stored in str.*
- auto `dispatch` (`const AddUser &cmd`) -> `void`  
*Visitor for the commands.*
- auto `dispatch` (`const Login &cmd`) -> `void`
- auto `dispatch` (`const Logout &cmd`) -> `void`
- auto `dispatch` (`const QueryProfile &cmd`) -> `void`
- auto `dispatch` (`const ModifyProfile &cmd`) -> `void`
- auto `dispatch` (`const AddTrain &cmd`) -> `void`
- auto `dispatch` (`const ReleaseTrain &cmd`) -> `void`
- auto `dispatch` (`const QueryTrain &cmd`) -> `void`
- auto `dispatch` (`const QueryTicket &cmd`) -> `void`
- auto `dispatch` (`const QueryTransfer &cmd`) -> `void`
- auto `dispatch` (`const BuyTicket &cmd`) -> `void`
- auto `dispatch` (`const QueryOrder &cmd`) -> `void`
- auto `dispatch` (`const RefundTicket &cmd`) -> `void`
- auto `dispatch` (`const Rollback &cmd`) -> `void`
- auto `dispatch` (`const Clean &cmd`) -> `void`
- auto `dispatch` (`const Exit &cmd`) -> `void`

### 5.2.1 Detailed Description

Classes and parsers for commands.

### 5.2.2 Typedef Documentation

**5.2.2.1 Command** using `ticket::command::Command` = typedef `Variant< AddUser, Login, Logout, QueryProfile, ModifyProfile, AddTrain, ReleaseTrain, QueryTrain, QueryTicket, QueryTransfer, BuyTicket, QueryOrder, RefundTicket, Rollback, Clean, Exit >`

### 5.2.3 Enumeration Type Documentation

**5.2.3.1 SortType** enum `ticket::command::SortType`

## Enumerator

kTime	
kCost	

## 5.2.4 Function Documentation

**5.2.4.1 dispatch()** [1/16] `auto ticket::command::dispatch (`  
`const AddTrain & cmd ) -> void`

**5.2.4.2 dispatch()** [2/16] `auto ticket::command::dispatch (`  
`const AddUser & cmd ) -> void`

Visitor for the commands.

The main function uses this visitor after parsing a command, to actually dispatch it. Overloads of operator() are callbacks of the commands.

The implementations are in the corresponding source files, not in [parser.cpp](#).

**5.2.4.3 dispatch()** [3/16] `auto ticket::command::dispatch (`  
`const BuyTicket & cmd ) -> void`

**5.2.4.4 dispatch()** [4/16] `auto ticket::command::dispatch (`  
`const Clean & cmd ) -> void`

**5.2.4.5 dispatch()** [5/16] `auto ticket::command::dispatch (`  
`const Exit & cmd ) -> void`

**5.2.4.6 dispatch()** [6/16] `auto ticket::command::dispatch (`  
`const Login & cmd ) -> void`

**5.2.4.7 dispatch()** [7/16] `auto ticket::command::dispatch (`  
`const Logout & cmd ) -> void`

**5.2.4.8 dispatch()** [8/16] `auto ticket::command::dispatch (`  
`const ModifyProfile & cmd ) -> void`

**5.2.4.9 dispatch()** [9/16] `auto ticket::command::dispatch (`  
`const QueryOrder & cmd ) -> void`

**5.2.4.10 dispatch()** [10/16] `auto ticket::command::dispatch (`  
`const QueryProfile & cmd ) -> void`

**5.2.4.11 dispatch()** [11/16] `auto ticket::command::dispatch (`  
`const QueryTicket & cmd ) -> void`

**5.2.4.12 dispatch()** [12/16] `auto ticket::command::dispatch (`  
`const QueryTrain & cmd ) -> void`

**5.2.4.13 dispatch()** [13/16] `auto ticket::command::dispatch (`  
`const QueryTransfer & cmd ) -> void`

**5.2.4.14 dispatch()** [14/16] `auto ticket::command::dispatch (`  
`const RefundTicket & cmd ) -> void`

**5.2.4.15 dispatch()** [15/16] `auto ticket::command::dispatch (`  
`const ReleaseTrain & cmd ) -> void`

**5.2.4.16 dispatch()** [16/16] `auto ticket::command::dispatch (`  
`const Rollback & cmd ) -> void`

**5.2.4.17 parse()** `auto ticket::command::parse ( std::string & str ) -> Result< Command, ParseException >`

parses the command stored in str.

this function is autogenerated.

## 5.3 ticket::file Namespace Reference

[File](#) utilities.

### Classes

- struct [Array](#)  
*An on-stack array with utility functions and bound checks.*
- class [BpTree](#)  
*an implementation of the B+ tree.*
- class [File](#)  
*A chunked file storage with manual garbage collection.*
- class [Index](#)  
*Class representing an index file.*
- class [Index< Varchar< maxLength >, Model, DataFile >](#)  
*Specialization of [Index](#) on [Varchar](#).*
- class [ManagedObject](#)  
*an opinionated utility base class for the objects to be stored.*
- struct [Set](#)  
*A sorted array with utility functions and bound checks.*
- struct [Varchar](#)  
*A wrapper for `const char *` with utility functions and type conversions.*

### Variables

- constexpr size\_t [kDefaultSzChunk](#) = 4096

### 5.3.1 Detailed Description

[File](#) utilities.

### 5.3.2 Variable Documentation

**5.3.2.1 kDefaultSzChunk** `constexpr size_t ticket::file::kDefaultSzChunk = 4096 [constexpr]`



## 5.4 ticket::rollback Namespace Reference

### Classes

- struct [AddTrain](#)
- struct [AddUser](#)
- struct [BuyTicket](#)
- struct [LogEntry](#)
- struct [ModifyProfile](#)
- struct [RefundTicket](#)
- struct [ReleaseTrain](#)

### Functions

- auto [dispatch](#) (const [AddUser](#) &log) -> void  
*Visitor for the log entries.*
- auto [dispatch](#) (const [ModifyProfile](#) &log) -> void
- auto [dispatch](#) (const [AddTrain](#) &log) -> void
- auto [dispatch](#) (const [ReleaseTrain](#) &log) -> void
- auto [dispatch](#) (const [BuyTicket](#) &log) -> void
- auto [dispatch](#) (const [RefundTicket](#) &log) -> void

### Variables

- [file::File](#) logEntries

### 5.4.1 Function Documentation

**5.4.1.1 dispatch()** [1/6] auto ticket::rollback::dispatch (  
const [AddTrain](#) & log ) -> void

**5.4.1.2 dispatch()** [2/6] auto ticket::rollback::dispatch (  
const [AddUser](#) & log ) -> void

Visitor for the log entries.

The implementations are in the corresponding source files, not in [rollback.cpp](#).

**5.4.1.3 dispatch()** [3/6] auto ticket::rollback::dispatch (  
const [BuyTicket](#) & log ) -> void

**5.4.1.4 dispatch()** [4/6] `auto ticket::rollback::dispatch (`  
`const ModifyProfile & log ) -> void`

**5.4.1.5 dispatch()** [5/6] `auto ticket::rollback::dispatch (`  
`const RefundTicket & log ) -> void`

**5.4.1.6 dispatch()** [6/6] `auto ticket::rollback::dispatch (`  
`const ReleaseTrain & log ) -> void`

## 5.4.2 Variable Documentation

**5.4.2.1 logEntries** `file::File ticket::rollback::logEntries [extern]`

## 5.5 ticket::Station Namespace Reference

### Typedefs

- using `Id` = `file::Varchar< 30 >`

### 5.5.1 Typedef Documentation

**5.5.1.1 Id** using `ticket::Station::Id` = typedef `file::Varchar<30>`

## 6 Class Documentation

### 6.1 ticket::command::AddTrain Struct Reference

```
#include <parser.h>
```

## Public Attributes

- std::string `id`
- int `stops`
- int `seats`
- Vector< std::string > `stations`
- Vector< int > `prices`
- Instant `departure`
- Vector< Duration > `durations`
- Vector< Duration > `stopoverTimes`
- Vector< Date > `dates`
- char `type`

### 6.1.1 Member Data Documentation

**6.1.1.1 dates** Vector<Date> ticket::command::AddTrain::dates

**6.1.1.2 departure** Instant ticket::command::AddTrain::departure

**6.1.1.3 durations** Vector<Duration> ticket::command::AddTrain::durations

**6.1.1.4 id** std::string ticket::command::AddTrain::id

**6.1.1.5 prices** Vector<int> ticket::command::AddTrain::prices

**6.1.1.6 seats** int ticket::command::AddTrain::seats

**6.1.1.7 stations** Vector<std::string> ticket::command::AddTrain::stations

**6.1.1.8 stopoverTimes** `Vector<Duration> ticket::command::AddTrain::stopoverTimes`

**6.1.1.9 stops** `int ticket::command::AddTrain::stops`

**6.1.1.10 type** `char ticket::command::AddTrain::type`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.2 ticket::rollback::AddTrain Struct Reference

```
#include <rollback.h>
```

### Public Attributes

- `int` [id](#)

### 6.2.1 Member Data Documentation

**6.2.1.1 id** `int ticket::rollback::AddTrain::id`

The documentation for this struct was generated from the following file:

- [src/rollback.h](#)

## 6.3 ticket::command::AddUser Struct Reference

```
#include <parser.h>
```

### Public Attributes

- `Optional< std::string >` [currentUser](#)
- `std::string` [username](#)
- `std::string` [password](#)
- `std::string` [name](#)
- `std::string` [email](#)
- `Optional< int >` [privilege](#)

### 6.3.1 Member Data Documentation

**6.3.1.1 currentUser** `Optional<std::string> ticket::command::AddUser::currentUser`

**6.3.1.2 email** `std::string ticket::command::AddUser::email`

**6.3.1.3 name** `std::string ticket::command::AddUser::name`

**6.3.1.4 password** `std::string ticket::command::AddUser::password`

**6.3.1.5 privilege** `Optional<int> ticket::command::AddUser::privilege`

**6.3.1.6 username** `std::string ticket::command::AddUser::username`

The documentation for this struct was generated from the following file:

- `src/parser.h`

## 6.4 ticket::rollback::AddUser Struct Reference

```
#include <rollback.h>
```

### Public Attributes

- `int id`

### 6.4.1 Member Data Documentation

#### 6.4.1.1 id int ticket::rollback::AddUser::id

The documentation for this struct was generated from the following file:

- [src/rollback.h](#)

## 6.5 ticket::file::Array< T, maxLength, Cmp > Struct Template Reference

An on-stack array with utility functions and bound checks.

```
#include <array.h>
```

### Public Member Functions

- auto [indexOf](#) (const T &element) -> size\_t  
*finds the index of element in the array.*
- auto [includes](#) (const T &element) -> bool  
*checks if the elements is included in the array.*
- auto [insert](#) (const T &element, size\_t offset) -> void  
*moves the elements after offset backwards, and inserts the element at the offset.*
- auto [remove](#) (const T &element) -> void  
*removes the element, and moves forward the elements after it.*
- auto [removeAt](#) (size\_t offset) -> void  
*removes the element at offset, and moves forward the elements after it.*
- auto [clear](#) () -> void  
*clears the array.*
- auto [copyFrom](#) (const [Array](#) &other, size\_t ixFrom, size\_t ixTo, size\_t count) -> void  
*copies a portion of another array to this.*
- auto [operator\[\]](#) (size\_t index) -> T &
- auto [operator\[\]](#) (size\_t index) const -> const T &
- auto [pop](#) () -> T  
*pops the last element.*
- auto [shift](#) () -> T  
*pops the first element.*
- auto [push](#) (const T &object) -> void  
*pushes after the last element.*
- auto [unshift](#) (const T &object) -> void  
*pushes before the first element.*
- template<typename Functor >  
auto [forEach](#) (const Functor &callback) -> T  
*calls the callback for each element in the array.*

### Public Attributes

- size\_t [length](#) = 0
- T [content](#) [maxLength]

### 6.5.1 Detailed Description

```
template<typename T, size_t maxLength, typename Cmp = Less<>>
struct ticket::file::Array< T, maxLength, Cmp >
```

An on-stack array with utility functions and bound checks.

The value type needs to be trivial.

### 6.5.2 Member Function Documentation

**6.5.2.1 clear()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`auto ticket::file::Array< T, maxLength, Cmp >::clear ( ) -> void [inline]`

clears the array.

**6.5.2.2 copyFrom()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`auto ticket::file::Array< T, maxLength, Cmp >::copyFrom (`  
    `const Array< T, maxLength, Cmp > & other,`  
    `size_t ixFrom,`  
    `size_t ixTo,`  
    `size_t count ) -> void [inline]`

copies a portion of another array to this.

**6.5.2.3 forEach()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`template<typename Functor >`  
`auto ticket::file::Array< T, maxLength, Cmp >::forEach (`  
    `const Functor & callback ) -> T [inline]`

calls the callback for each element in the array.

**6.5.2.4 includes()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`auto ticket::file::Array< T, maxLength, Cmp >::includes (`  
    `const T & element ) -> bool [inline]`

checks if the elements is included in the array.

**6.5.2.5 indexOf()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::indexOf (   
 const T & element ) -> size_t [inline]`

finds the index of element in the array.

**6.5.2.6 insert()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::insert (   
 const T & element,  
 size_t offset ) -> void [inline]`

moves the elements after offset backwards, and inserts the element at the offset.

**6.5.2.7 operator[]()** [1/2] `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::operator[] (   
 size_t index ) -> T & [inline]`

**6.5.2.8 operator[]()** [2/2] `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::operator[] (   
 size_t index ) const -> const T & [inline]`

**6.5.2.9 pop()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::pop ( ) -> T [inline]`

pops the last element.

**6.5.2.10 push()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::push (   
 const T & object ) -> void [inline]`

pushes after the last element.

**6.5.2.11 remove()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::remove (   
 const T & element ) -> void [inline]`

removes the element, and moves forward the elements after it.



**6.5.2.12 removeAt()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::removeAt (   
 size_t offset ) -> void [inline]`

removes the element at offset, and moves forward the elements after it.

**6.5.2.13 shift()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::shift ( ) -> T [inline]`

pops the first element.

**6.5.2.14 unshift()** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
auto ticket::file::Array< T, maxLength, Cmp >::unshift (   
 const T & object ) -> void [inline]`

pushes before the first element.

### 6.5.3 Member Data Documentation

**6.5.3.1 content** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
T ticket::file::Array< T, maxLength, Cmp >::content[maxLength]`

**6.5.3.2 length** `template<typename T , size_t maxLength, typename Cmp = Less<>>  
size_t ticket::file::Array< T, maxLength, Cmp >::length = 0`

The documentation for this struct was generated from the following file:

- lib/file/array.h

## 6.6 ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk > Class Template Reference

an implementation of the B+ tree.

```
#include <bptree.h>
```

## Public Member Functions

- [BpTree](#) (const char \*filename)  
*constructs a B+ tree on the given file.*
- auto [insert](#) (const KeyType &key, const ValueType &value) -> void  
*inserts a key-value pair into the tree.*
- auto [remove](#) (const KeyType &key, const ValueType &value) -> void  
*removes a key-value pair from the tree.*
- auto [findOne](#) (const KeyType &key) -> [Optional](#)< ValueType >  
*finds the first entry with the given key.*
- auto [findMany](#) (const KeyType &key) -> [Vector](#)< ValueType >  
*finds all entries with the given key.*
- auto [findAll](#) () -> [Vector](#)< [ticket::Pair](#)< KeyType, ValueType > >  
*finds all entries.*
- auto [includes](#) (const KeyType &key, const ValueType &value) -> bool  
*checks if the given key-value pair exists in the tree.*
- auto [getMeta](#) () -> Meta  
*gets user-provided metadata.*
- auto [setMeta](#) (const Meta &meta) -> void  
*sets user-provided metadata.*
- auto [clearCache](#) () -> void  
*clears the cache of the underlying file.*

### 6.6.1 Detailed Description

```
template<typename KeyType, typename ValueType, typename CmpKey = Less<>, typename CmpValue = Less<>, typename
Meta = Unit, size_t szChunk = kDefaultSzChunk>
class ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >
```

an implementation of the B+ tree.

it stores key and value together in order to support duplicate keys.

constraints: KeyType and ValueType need to be comparable.

### 6.6.2 Constructor & Destructor Documentation

**6.6.2.1 BpTree()**

```
template<typename KeyType , typename ValueType , typename CmpKey = Less<>,
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::BpTree (
    const char * filename ) [inline]
```

constructs a B+ tree on the given file.

### 6.6.3 Member Function Documentation

**6.6.3.1 clearCache()** `template<typename KeyType , typename ValueType , typename CmpKey = Less<>,  
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::clearCache ( )  
-> void [inline]`

clears the cache of the underlying file.

you may need to call this method periodically to avoid using up too much memory.

**6.6.3.2 findAll()** `template<typename KeyType , typename ValueType , typename CmpKey = Less<>,  
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::findAll ( )  
-> Vector<ticket::Pair<KeyType, ValueType>> [inline]`

finds all entries.

**6.6.3.3 findMany()** `template<typename KeyType , typename ValueType , typename CmpKey = Less<>,  
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::findMany (   
const KeyType & key ) -> Vector<ValueType> [inline]`

finds all entries with the given key.

**6.6.3.4 findOne()** `template<typename KeyType , typename ValueType , typename CmpKey = Less<>,  
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::findOne (   
const KeyType & key ) -> Optional<ValueType> [inline]`

finds the first entry with the given key.

**6.6.3.5 getMeta()** `template<typename KeyType , typename ValueType , typename CmpKey = Less<>,  
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::getMeta ( )  
-> Meta [inline]`

gets user-provided metadata.

**6.6.3.6 includes()** `template<typename KeyType , typename ValueType , typename CmpKey = Less<>,  
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::includes (   
const KeyType & key,  
const ValueType & value ) -> bool [inline]`

checks if the given key-value pair exists in the tree.

```
6.6.3.7 insert() template<typename KeyType , typename ValueType , typename CmpKey = Less<>,
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::insert (
    const KeyType & key,
    const ValueType & value ) -> void    [inline]
```

inserts a key-value pair into the tree.

duplicate keys is supported, though duplicate key-value pair leads to undefined behavior, and may lead to an invalid tree.

```
6.6.3.8 remove() template<typename KeyType , typename ValueType , typename CmpKey = Less<>,
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::remove (
    const KeyType & key,
    const ValueType & value ) -> void    [inline]
```

removes a key-value pair from the tree.

you must ensure that the entry is indeed in the tree. removing an nonexistent entry may lead to an invalid tree.

```
6.6.3.9 setMeta() template<typename KeyType , typename ValueType , typename CmpKey = Less<>,
typename CmpValue = Less<>, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
auto ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >::setMeta (
    const Meta & meta ) -> void    [inline]
```

sets user-provided metadata.

The documentation for this class was generated from the following file:

- [lib/file/bptree.h](#)

## 6.7 ticket::command::BuyTicket Struct Reference

```
#include <parser.h>
```

### Public Attributes

- std::string [currentUser](#)
- std::string [train](#)
- [Date](#) [date](#)
- int [seats](#)
- std::string [from](#)
- std::string [to](#)
- bool [queue](#) = false

### 6.7.1 Member Data Documentation

**6.7.1.1 currentUser** `std::string ticket::command::BuyTicket::currentUser`

**6.7.1.2 date** `Date ticket::command::BuyTicket::date`

**6.7.1.3 from** `std::string ticket::command::BuyTicket::from`

**6.7.1.4 queue** `bool ticket::command::BuyTicket::queue = false`

**6.7.1.5 seats** `int ticket::command::BuyTicket::seats`

**6.7.1.6 to** `std::string ticket::command::BuyTicket::to`

**6.7.1.7 train** `std::string ticket::command::BuyTicket::train`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.8 ticket::rollback::BuyTicket Struct Reference

```
#include <rollback.h>
```

### Public Attributes

- `int id`

#### 6.8.1 Member Data Documentation

**6.8.1.1 id** `int ticket::rollback::BuyTicket::id`

The documentation for this struct was generated from the following file:

- [src/rollback.h](#)

## 6.9 ticket::command::Clean Struct Reference

```
#include <parser.h>
```

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.10 ticket::Cmp< Lt > Class Template Reference

Comparison utilities.

```
#include <utility.h>
```

### Public Member Functions

- `template<typename T, typename U >`  
`auto equals (const T &lhs, const U &rhs) -> bool`
- `template<typename T, typename U >`  
`auto ne (const T &lhs, const U &rhs) -> bool`
- `template<typename T, typename U >`  
`auto lt (const T &lhs, const U &rhs) -> bool`
- `template<typename T, typename U >`  
`auto gt (const T &lhs, const U &rhs) -> bool`
- `template<typename T, typename U >`  
`auto leq (const T &lhs, const U &rhs) -> bool`
- `template<typename T, typename U >`  
`auto geq (const T &lhs, const U &rhs) -> bool`

### 6.10.1 Detailed Description

```
template<typename Lt>
class ticket::Cmp< Lt >
```

Comparison utilities.

### 6.10.2 Member Function Documentation

**6.10.2.1 equals()** `template<typename Lt >`  
`template<typename T , typename U >`  
`auto ticket::Cmp< Lt >::equals (`  
    `const T & lhs,`  
    `const U & rhs ) -> bool   [inline]`

**6.10.2.2 geq()** `template<typename Lt >`  
`template<typename T , typename U >`  
`auto ticket::Cmp< Lt >::geq (`  
    `const T & lhs,`  
    `const U & rhs ) -> bool   [inline]`

**6.10.2.3 gt()** `template<typename Lt >`  
`template<typename T , typename U >`  
`auto ticket::Cmp< Lt >::gt (`  
    `const T & lhs,`  
    `const U & rhs ) -> bool   [inline]`

**6.10.2.4 leq()** `template<typename Lt >`  
`template<typename T , typename U >`  
`auto ticket::Cmp< Lt >::leq (`  
    `const T & lhs,`  
    `const U & rhs ) -> bool   [inline]`

**6.10.2.5 lt()** `template<typename Lt >`  
`template<typename T , typename U >`  
`auto ticket::Cmp< Lt >::lt (`  
    `const T & lhs,`  
    `const U & rhs ) -> bool   [inline]`

**6.10.2.6 ne()** `template<typename Lt >`  
`template<typename T , typename U >`  
`auto ticket::Cmp< Lt >::ne (`  
    `const T & lhs,`  
    `const U & rhs ) -> bool   [inline]`

The documentation for this class was generated from the following file:

- [lib/utility.h](#)

## 6.11 ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator Class Reference

```
#include <hashmap.h>
```

### Public Types

- using [difference\\_type](#) = std::ptrdiff\_t
- using [value\\_type](#) = const [HashMap::value\\_type](#)
- using [pointer](#) = [value\\_type](#) \*
- using [reference](#) = [value\\_type](#) &
- using [iterator\\_category](#) = std::output\_iterator\_tag

### Public Member Functions

- [const\\_iterator](#) ()=default
- [const\\_iterator](#) (const ListNode \*node, const [HashMap](#) \*home)
- [const\\_iterator](#) (const [iterator](#) &other)
- auto [operator++](#) (int) -> [const\\_iterator](#)
- auto [operator++](#) () -> [const\\_iterator](#) &
- auto [operator--](#) (int) -> [const\\_iterator](#)
- auto [operator--](#) () -> [const\\_iterator](#) &
- auto [operator\\*](#) () const -> [reference](#)
- auto [operator==](#) (const [iterator](#) &rhs) const -> bool
- auto [operator==](#) (const [const\\_iterator](#) &rhs) const -> bool
- auto [operator!=](#) (const [iterator](#) &rhs) const -> bool
- auto [operator!=](#) (const [const\\_iterator](#) &rhs) const -> bool
- auto [operator->](#) () const noexcept -> [pointer](#)

### Friends

- class [iterator](#)
- class [HashMap](#)

#### 6.11.1 Member Typedef Documentation

**6.11.1.1 difference\_type** `template<typename Key , typename Value , typename Hash = std::hash<↵  
Key>, typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::difference_type = std::↵  
::ptrdiff_t`

**6.11.1.2 iterator\_category** `template<typename Key , typename Value , typename Hash = std::hash<↵  
Key>, typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::iterator_category = std::↵  
::output_iterator_tag`



**6.11.1.3 pointer** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::pointer = value\_type *`

**6.11.1.4 reference** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::reference = value\_type &`

**6.11.1.5 value\_type** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::value_type = const HashMap::value\_type`

## 6.11.2 Constructor & Destructor Documentation

**6.11.2.1 const\_iterator() [1/3]** `template<typename Key , typename Value , typename Hash = std↵  
::hash<Key> , typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::const_iterator ( ) [default]`

**6.11.2.2 const\_iterator() [2/3]** `template<typename Key , typename Value , typename Hash = std↵  
::hash<Key> , typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::const_iterator (   
const ListNode * node,  
const HashMap * home ) [inline]`

**6.11.2.3 const\_iterator() [3/3]** `template<typename Key , typename Value , typename Hash = std↵  
::hash<Key> , typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::const_iterator (   
const iterator & other ) [inline]`

## 6.11.3 Member Function Documentation

**6.11.3.1 operator!=( ) [1/2]** `template<typename Key , typename Value , typename Hash = std↵  
::hash<Key> , typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator!= (   
const const\_iterator & rhs ) const -> bool [inline]`

**6.11.3.2 operator!=( ) [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator!= (   
const iterator & rhs ) const -> bool [inline]`

**6.11.3.3 operator\*() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator* ( ) const -> reference [inline]`

**6.11.3.4 operator++() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator++ ( ) -> const_iterator & [inline]`

**6.11.3.5 operator++() [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator++ (   
int ) -> const_iterator [inline]`

**6.11.3.6 operator--() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator-- ( ) -> const_iterator & [inline]`

**6.11.3.7 operator--() [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator-- (   
int ) -> const_iterator [inline]`

**6.11.3.8 operator->() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator-> ( ) const -> pointer [inline], [noexcept]`

```
6.11.3.9 operator==( [1/2] template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator== (
    const const_iterator & rhs ) const -> bool    [inline]
```

```
6.11.3.10 operator==( [2/2] template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>
auto ticket::HashMap< Key, Value, Hash, Equal >::const_iterator::operator== (
    const iterator & rhs ) const -> bool    [inline]
```

## 6.11.4 Friends And Related Function Documentation

```
6.11.4.1 HashMap template<typename Key , typename Value , typename Hash = std::hash<Key>,
typename Equal = std::equal_to<Key>>
friend class HashMap    [friend]
```

```
6.11.4.2 iterator template<typename Key , typename Value , typename Hash = std::hash<Key>,
typename Equal = std::equal_to<Key>>
friend class iterator    [friend]
```

The documentation for this class was generated from the following file:

- [lib/hashmap.h](#)

## 6.12 **ticket::Vector< T >::const\_iterator** Class Reference

```
#include <vector.h>
```

### Public Types

- using [difference\\_type](#) = std::ptrdiff\_t
- using [value\\_type](#) = T
- using [pointer](#) = T \*
- using [reference](#) = T &
- using [iterator\\_category](#) = std::output\_iterator\_tag

## Public Member Functions

- auto [operator+](#) (const int &n) const -> [const\\_iterator](#)
- auto [operator-](#) (const int &n) const -> [const\\_iterator](#)
- auto [operator-](#) (const [const\\_iterator](#) &rhs) const -> int
- auto [operator+=](#) (const int &n) -> [const\\_iterator](#) &
- auto [operator-=](#) (const int &n) -> [const\\_iterator](#) &
- auto [operator++](#) (int) const -> [const\\_iterator](#)
- auto [operator++](#) () -> [const\\_iterator](#) &
- auto [operator--](#) (int) const -> [const\\_iterator](#)
- auto [operator--](#) () -> [const\\_iterator](#) &
- auto [operator\\*](#) () const -> const T &
- auto [operator==](#) (const [iterator](#) &rhs) const -> bool
- auto [operator==](#) (const [const\\_iterator](#) &rhs) const -> bool
- auto [operator!=](#) (const [iterator](#) &rhs) const -> bool
- auto [operator!=](#) (const [const\\_iterator](#) &rhs) const -> bool
- auto [operator<](#) (const [iterator](#) &rhs) const -> bool
- auto [operator<](#) (const [const\\_iterator](#) &rhs) const -> bool

## Friends

- class [iterator](#)
- class [Vector](#)

### 6.12.1 Member Typedef Documentation

**6.12.1.1 difference\_type** `template<typename T >`  
`using ticket::Vector< T >::const_iterator::difference_type = std::ptrdiff_t`

**6.12.1.2 iterator\_category** `template<typename T >`  
`using ticket::Vector< T >::const_iterator::iterator_category = std::output_iterator_tag`

**6.12.1.3 pointer** `template<typename T >`  
`using ticket::Vector< T >::const_iterator::pointer = T *`

**6.12.1.4 reference** `template<typename T >`  
`using ticket::Vector< T >::const_iterator::reference = T &`

**6.12.1.5 value\_type** `template<typename T >`  
`using ticket::Vector< T >::const_iterator::value_type = T`

## 6.12.2 Member Function Documentation

**6.12.2.1 operator!=(())** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::const_iterator::operator!= (`  
`const const_iterator & rhs ) const -> bool [inline]`

**6.12.2.2 operator!=(())** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::const_iterator::operator!= (`  
`const iterator & rhs ) const -> bool [inline]`

**6.12.2.3 operator\*()** `template<typename T >`  
`auto ticket::Vector< T >::const_iterator::operator* ( ) const -> const T & [inline]`

**6.12.2.4 operator+()** `template<typename T >`  
`auto ticket::Vector< T >::const_iterator::operator+ (`  
`const int & n ) const -> const_iterator [inline]`

**6.12.2.5 operator++()** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::const_iterator::operator++ ( ) -> const_iterator & [inline]`

**6.12.2.6 operator++()** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::const_iterator::operator++ (`  
`int ) const -> const_iterator [inline]`

**6.12.2.7 operator+=()** `template<typename T >`  
`auto ticket::Vector< T >::const_iterator::operator+= (`  
`const int & n ) -> const_iterator & [inline]`

**6.12.2.8 operator-()** [1/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator- (   
const const\_iterator & rhs ) const -> int [inline]

**6.12.2.9 operator-()** [2/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator- (   
const int & n ) const -> const\_iterator [inline]

**6.12.2.10 operator--()** [1/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator-- ( ) -> const\_iterator & [inline]

**6.12.2.11 operator--()** [2/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator-- (   
int ) const -> const\_iterator [inline]

**6.12.2.12 operator-=()** template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator-= (   
const int & n ) -> const\_iterator & [inline]

**6.12.2.13 operator<()** [1/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator< (   
const const\_iterator & rhs ) const -> bool [inline]

**6.12.2.14 operator<()** [2/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator< (   
const iterator & rhs ) const -> bool [inline]

**6.12.2.15 operator==()** [1/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator== (   
const const\_iterator & rhs ) const -> bool [inline]

**6.12.2.16 operator==()** [2/2] template<typename T >  
auto ticket::Vector< T >::const\_iterator::operator== (   
const iterator & rhs ) const -> bool [inline]

### 6.12.3 Friends And Related Function Documentation

**6.12.3.1 iterator** `template<typename T >`  
`friend class iterator [friend]`

**6.12.3.2 Vector** `template<typename T >`  
`friend class Vector [friend]`

The documentation for this class was generated from the following file:

- [lib/vector.h](#)

### 6.13 ticket::Date Class Reference

Class representing a date between 2021-06-01 and 2021-08-31 (inclusive).

```
#include <datetime.h>
```

#### Public Member Functions

- [Date](#) ()=default
- [Date](#) (int [month](#), int [date](#))
- [Date](#) (const char \*str)  
*constructs a [Date](#) from a MM-DD format string.*
- auto [month](#) () const -> int  
*gets the month of the [Date](#). (Fri Jun 04 2021 -> 6)*
- auto [date](#) () const -> int  
*gets the date of the [Date](#). (Fri Jun 04 2021 -> 4)*
- [operator std::string](#) () const  
*gets a MM-DD representation of the [Date](#).*
- auto [operator+](#) (int dt) const -> [Date](#)  
*calculates a date dt days after this [Date](#). (06-04 + 3 == 06-07)*
- auto [operator-](#) (int dt) const -> [Date](#)  
*calculates a date dt days before this [Date](#). (06-04 - 3 == 06-01)*
- auto [operator-](#) ([Date](#) rhs) const -> int  
*calculates the difference between two Dates. (06-04 - 06-01 == 3)*
- auto [operator<](#) (const [Date](#) &rhs) const -> bool
- auto [inRange](#) ([Date](#) begin, [Date](#) end) const -> bool  
*checks if this [Date](#) is in the given range (inclusive).*

#### 6.13.1 Detailed Description

Class representing a date between 2021-06-01 and 2021-08-31 (inclusive).

## 6.13.2 Constructor & Destructor Documentation

**6.13.2.1 Date()** [1/3] `ticket::Date::Date ( ) [default]`

**6.13.2.2 Date()** [2/3] `ticket::Date::Date (`  
    `int month,`  
    `int date )`

**6.13.2.3 Date()** [3/3] `ticket::Date::Date (`  
    `const char * str ) [explicit]`

constructs a [Date](#) from a MM-DD format string.

it is an undefined behavior if the string is not in MM-DD format, is nullptr, or points to invalid memory.

## 6.13.3 Member Function Documentation

**6.13.3.1 date()** `auto ticket::Date::date ( ) const -> int`

gets the date of the [Date](#). (Fri Jun 04 2021 -> 4)

**6.13.3.2 inRange()** `auto ticket::Date::inRange (`  
    [Date](#) *begin*,  
    [Date](#) *end* ) `const -> bool`

checks if this [Date](#) is in the given range (inclusive).

**6.13.3.3 month()** `auto ticket::Date::month ( ) const -> int`

gets the month of the [Date](#). (Fri Jun 04 2021 -> 6)



**6.13.3.4 operator std::string()** `ticket::Date::operator std::string ( ) const`

gets a MM-DD representation of the [Date](#).

**6.13.3.5 operator+()** `auto ticket::Date::operator+ (   
int dt ) const -> Date`

calculates a date dt days after this [Date](#). (06-04 + 3 == 06-07)

**6.13.3.6 operator-() [1/2]** `auto ticket::Date::operator- (   
Date rhs ) const -> int`

calculates the difference between two Dates. (06-04 - 06-01 == 3)

**6.13.3.7 operator-() [2/2]** `auto ticket::Date::operator- (   
int dt ) const -> Date`

calculates a date dt days before this [Date](#). (06-04 - 3 == 06-01)

**6.13.3.8 operator<()** `auto ticket::Date::operator< (   
const Date & rhs ) const -> bool`

The documentation for this class was generated from the following file:

- [lib/datetime.h](#)

## 6.14 ticket::Duration Class Reference

Class representing a length of timespan.

```
#include <datetime.h>
```

## Public Member Functions

- [Duration](#) ()=default
- [Duration](#) (int hour, int minute)
- [Duration](#) (int minutes)
- [Duration](#) (const char \*str)  
*constructs a [Duration](#) from an HH:MM format string.*
- auto [hours](#) () const -> int  
*gets the hour part of the duration, may be negative.*
- auto [minutes](#) () const -> int  
*gets the minute part of the duration, may be negative.*
- auto [totalMinutes](#) () const -> int  
*gets how many minutes are there in this [Duration](#).*
- auto [operator+](#) ([Duration](#) dt) const -> [Duration](#)
- auto [operator-](#) ([Duration](#) dt) const -> [Duration](#)
- auto [operator-](#) () const -> [Duration](#)  
*negates the [Duration](#).*
- auto [operator<](#) (const [Duration](#) &rhs) const -> bool

### 6.14.1 Detailed Description

Class representing a length of timespan.

The length may be positive, zero or negative.

Not to be confused with [Instant](#), which is a fixed point of time. For example, 02:10 as in “brewing the tea takes 02:10” is a duration, while 02:10 as in “it’s 02:10 now, go to sleep right now” is an instant.

### 6.14.2 Constructor & Destructor Documentation

**6.14.2.1 [Duration\(\)](#) [1/4]** `ticket::Duration::Duration ( ) [default]`

**6.14.2.2 [Duration\(\)](#) [2/4]** `ticket::Duration::Duration (`  
`int hour,`  
`int minute )`

**6.14.2.3 [Duration\(\)](#) [3/4]** `ticket::Duration::Duration (`  
`int minutes ) [explicit]`

**6.14.2.4 Duration()** [4/4] `ticket::Duration::Duration (`  
`const char * str ) [explicit]`

constructs a [Duration](#) from an HH:MM format string.

### 6.14.3 Member Function Documentation

**6.14.3.1 hours()** `auto ticket::Duration::hours ( ) const -> int`

gets the hour part of the duration, may be negative.

**6.14.3.2 minutes()** `auto ticket::Duration::minutes ( ) const -> int`

gets the minute part of the duration, may be negative.

**6.14.3.3 operator+()** `auto ticket::Duration::operator+ (`  
`Duration dt ) const -> Duration`

**6.14.3.4 operator-()** [1/2] `auto ticket::Duration::operator- ( ) const -> Duration`

negates the [Duration](#).

**6.14.3.5 operator-()** [2/2] `auto ticket::Duration::operator- (`  
`Duration dt ) const -> Duration`

**6.14.3.6 operator<()** `auto ticket::Duration::operator< (`  
`const Duration & rhs ) const -> bool`

**6.14.3.7 totalMinutes()** `auto ticket::Duration::totalMinutes ( ) const -> int`

gets how many minutes are there in this [Duration](#).

The documentation for this class was generated from the following file:

- [lib/datetime.h](#)

## 6.15 ticket::Train::Edge Struct Reference

```
#include <train.h>
```

### Public Attributes

- int [price](#)
- [Instant](#) [departure](#)
- [Instant](#) [arrival](#)

### 6.15.1 Member Data Documentation

**6.15.1.1 arrival** [Instant](#) `ticket::Train::Edge::arrival`

**6.15.1.2 departure** [Instant](#) `ticket::Train::Edge::departure`

**6.15.1.3 price** `int ticket::Train::Edge::price`

The documentation for this struct was generated from the following file:

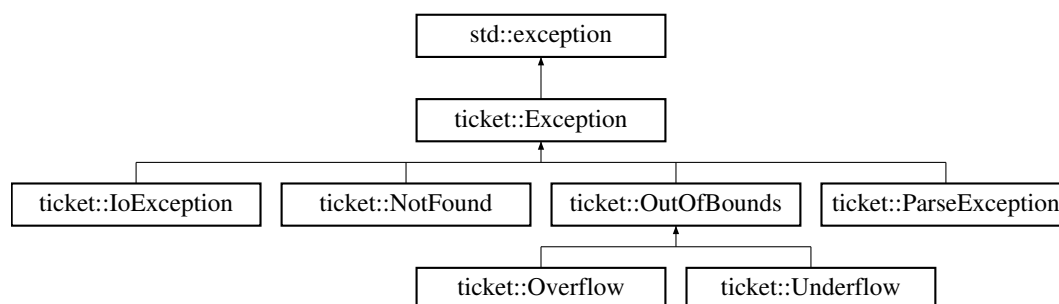
- [src/train.h](#)

## 6.16 ticket::Exception Class Reference

The base exception class.

```
#include <exception.h>
```

Inheritance diagram for `ticket::Exception`:



## Public Member Functions

- [Exception](#) ()=default
- [Exception](#) (const char \*[what](#))
- virtual [~Exception](#) ()=default
- virtual auto [what](#) () const noexcept -> const char \*  
*returns a human-readable description of the exception.*

### 6.16.1 Detailed Description

The base exception class.

### 6.16.2 Constructor & Destructor Documentation

**6.16.2.1 [Exception\(\)](#)** [1/2] `ticket::Exception::Exception ( )` [default]

**6.16.2.2 [Exception\(\)](#)** [2/2] `ticket::Exception::Exception (`  
`const char * what )` [inline]

**6.16.2.3 [~Exception\(\)](#)** `virtual ticket::Exception::~~Exception ( )` [virtual], [default]

### 6.16.3 Member Function Documentation

**6.16.3.1 [what\(\)](#)** `virtual auto ticket::Exception::what ( ) const -> const char *` [inline],  
[virtual], [noexcept]

returns a human-readable description of the exception.

The documentation for this class was generated from the following file:

- lib/[exception.h](#)

## 6.17 ticket::command::Exit Struct Reference

```
#include <parser.h>
```

The documentation for this struct was generated from the following file:

- src/[parser.h](#)

## 6.18 ticket::file::File< Meta, szChunk > Class Template Reference

A chunked file storage with manual garbage collection.

```
#include <file.h>
```

### Public Member Functions

- `template<typename Functor >`  
`File` (const char \*filename, const Functor &initializer)  
*initializes the file at filename.*
- `File` (const char \*filename)
- `~File` ()
- `auto get` (void \*buf, size\_t index, size\_t n) -> void  
*read n bytes at index into buf.*
- `auto set` (const void \*buf, size\_t index, size\_t n) -> void  
*write n bytes at index from buf.*
- `auto push` (const void \*buf, size\_t n) -> size\_t
- `auto remove` (size\_t index) -> void
- `auto getMeta` () -> Meta  
*gets user-provided metadata.*
- `auto setMeta` (const Meta &user) -> void  
*sets user-provided metadata.*
- `auto clearCache` () -> void  
*clears the cache.*

### 6.18.1 Detailed Description

```
template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
class ticket::file::File< Meta, szChunk >
```

A chunked file storage with manual garbage collection.

It is of chunk size of szChunk and has cache powered by [HashMap](#).

### 6.18.2 Constructor & Destructor Documentation

**6.18.2.1 File()** [1/2] `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>`  
`template<typename Functor >`  
`ticket::file::File< Meta, szChunk >::File` (  
    const char \* filename,  
    const Functor & initializer ) [inline]

initializes the file at filename.

it is not thread-safe.

**Parameters**

<i>filename</i>	the file to open
<i>initializer</i>	callback called on the creation of the file, when the file is empty.

**6.18.2.2 File()** [2/2] `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
ticket::file::File< Meta, szChunk >::File (  
 const char * filename ) [inline]`

**6.18.2.3 ~File()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
ticket::file::File< Meta, szChunk >::~~File ( ) [inline]`

**6.18.3 Member Function Documentation**

**6.18.3.1 clearCache()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::File< Meta, szChunk >::clearCache ( ) -> void [inline]`

clears the cache.

**6.18.3.2 get()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::File< Meta, szChunk >::get (  
 void * buf,  
 size_t index,  
 size_t n ) -> void [inline]`

read n bytes at index into buf.

**6.18.3.3 getMeta()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>  
auto ticket::file::File< Meta, szChunk >::getMeta ( ) -> Meta [inline]`

gets user-provided metadata.

**6.18.3.4 push()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>`  
`auto ticket::file::File< Meta, szChunk >::push (`  
`const void * buf,`  
`size_t n ) -> size_t   [inline]`

**Returns**

the stored index of the object

**6.18.3.5 remove()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>`  
`auto ticket::file::File< Meta, szChunk >::remove (`  
`size_t index ) -> void   [inline]`

**6.18.3.6 set()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>`  
`auto ticket::file::File< Meta, szChunk >::set (`  
`const void * buf,`  
`size_t index,`  
`size_t n ) -> void   [inline]`

write n bytes at index from buf.

**6.18.3.7 setMeta()** `template<typename Meta = Unit, size_t szChunk = kDefaultSzChunk>`  
`auto ticket::file::File< Meta, szChunk >::setMeta (`  
`const Meta & user ) -> void   [inline]`

sets user-provided metadata.

The documentation for this class was generated from the following file:

- lib/file/file.h

## 6.19 ticket::HashMap< Key, Value, Hash, Equal > Class Template Reference

An unordered hash-based map.

```
#include <hashmap.h>
```

**Classes**

- class [const\\_iterator](#)
- class [iterator](#)



## Public Types

- using `value_type` = `Pair`< const Key, Value >

## Public Member Functions

- `HashMap` ()=default
- `HashMap` (const `HashMap` &other)
- auto `operator=` (const `HashMap` &other) -> `HashMap` &
- `~HashMap` ()
- auto `at` (const Key &key) -> Value &
- auto `at` (const Key &key) const -> const Value &
- auto `operator[]` (const Key &key) -> Value &
- auto `operator[]` (const Key &key) const -> const Value &
- *behave like `at()` throw `index_out_of_bound` if such key does not exist.*
- auto `begin` () -> `iterator`
- *return a iterator to the beginning*
- auto `cbegin` () const -> `const_iterator`
- auto `end` () -> `iterator`
- *return a iterator to the end*
- auto `cend` () const -> `const_iterator`
- auto `empty` () const -> bool
- *checks whether the container is empty*
- auto `size` () const -> `size_t`
- *returns the number of elements.*
- auto `clear` () -> void
- *clears the contents*
- auto `insert` (const `value_type` &value) -> `Pair`< `iterator`, bool >
- auto `erase` (`iterator` pos) -> void
- auto `count` (const Key &key) const -> `size_t`
- auto `find` (const Key &key) -> `iterator`
- auto `find` (const Key &key) const -> `const_iterator`

### 6.19.1 Detailed Description

```
template<typename Key, typename Value, typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>
class ticket::HashMap< Key, Value, Hash, Equal >
```

An unordered hash-based map.

In `HashMap`, iteration ordering is differ from `map`, which is the order in which keys were inserted into the map. You should maintain a doubly-linked list running through all of its entries to keep the correct iteration order.

Note that insertion order is not affected if a key is re-inserted into the map.

### 6.19.2 Member Typedef Documentation

**6.19.2.1 value\_type** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::value_type = Pair<const Key, Value>`

## 6.19.3 Constructor & Destructor Documentation

**6.19.3.1 HashMap() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::HashMap ( ) [default]`

**6.19.3.2 HashMap() [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::HashMap (   
const HashMap< Key, Value, Hash, Equal > & other ) [inline]`

**6.19.3.3 ~HashMap()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::~HashMap ( ) [inline]`

## 6.19.4 Member Function Documentation

**6.19.4.1 at() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::at (   
const Key & key ) -> Value & [inline]`

access specified element with bounds checking Returns a reference to the mapped value of the element with key equivalent to key. If no such element exists, an exception of type 'index\_out\_of\_bound'

**6.19.4.2 at() [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::at (   
const Key & key ) const -> const Value & [inline]`

**6.19.4.3 begin()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::begin ( ) -> iterator [inline]`

return a iterator to the beginning

**6.19.4.4 cbegin()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::cbegin ( ) const -> const_iterator [inline]`

**6.19.4.5 cend()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::cend ( ) const -> const_iterator [inline]`

**6.19.4.6 clear()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::clear ( ) -> void [inline]`

clears the contents

**6.19.4.7 count()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::count (   
const Key & key ) const -> size_t [inline]`

Returns the number of elements with key that compares equivalent to the specified argument, which is either 1 or 0 since this container does not allow duplicates.

**6.19.4.8 empty()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::empty ( ) const -> bool [inline]`

checks whether the container is empty

**6.19.4.9 end()** `template<typename Key , typename Value , typename Hash = std::hash<Key> , typename  
Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::end ( ) -> iterator [inline]`

return a iterator to the end

**6.19.4.10 erase()** `template<typename Key , typename Value , typename Hash = std::hash<Key> ,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::erase (   
    iterator pos ) -> void [inline]`

erase the element at pos. throw if pos pointed to a bad element (pos == this->end() || pos points an element out of this)

**6.19.4.11 find() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key> ,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::find (   
    const Key & key ) -> iterator [inline]`

Finds an element with key equivalent to key. key value of the element to search for. Iterator to an element with key equivalent to key. If no such element is found, past-the-end (see end()) iterator is returned.

**6.19.4.12 find() [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key> ,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::find (   
    const Key & key ) const -> const_iterator [inline]`

**6.19.4.13 insert()** `template<typename Key , typename Value , typename Hash = std::hash<Key> ,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::insert (   
    const value_type & value ) -> Pair<iterator, bool> [inline]`

insert an element. return a pair, the first of the pair is the iterator to the new element (or the element that prevented the insertion), the second one is true if insert successfully, or false.

**6.19.4.14 operator=()** `template<typename Key , typename Value , typename Hash = std::hash<Key> ,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::operator= (   
    const HashMap< Key, Value, Hash, Equal > & other ) -> HashMap & [inline]`

**6.19.4.15 operator[]() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key> ,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::operator[] (   
    const Key & key ) -> Value & [inline]`

access specified element Returns a reference to the value that is mapped to a key equivalent to key, performing an insertion if such key does not already exist.

**6.19.4.16 operator[]() [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<Key> ,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::operator[] (   
    const Key & key ) const -> const Value & [inline]`

behave like at() throw index\_out\_of\_bound if such key does not exist.

```

6.19.4.17 size() template<typename Key , typename Value , typename Hash = std::hash<Key>,
typename Equal = std::equal_to<Key>>
auto ticket::HashMap< Key, Value, Hash, Equal >::size ( ) const -> size_t    [inline]

```

returns the number of elements.

The documentation for this class was generated from the following file:

- lib/hashmap.h

## 6.20 ticket::file::Index< Key, Model, DataFile > Class Template Reference

Class representing an index file.

```
#include <index.h>
```

### Public Member Functions

- [Index](#) (Key Model::\*ptr, const char \*filename, DataFile &datafile)  
*initializes the index.*
- auto [insert](#) (const Model &model) -> void  
*inserts an object into the index.*
- auto [remove](#) (const Model &model) -> void  
*removes an object from the index.*
- auto [findOne](#) (const Key &key) -> [Optional](#)< Model >  
*finds one Model in the index.*
- auto [findOneId](#) (const Key &key) -> [Optional](#)< int >  
*finds one identifier in the index.*
- auto [findMany](#) (const Key &key) -> [Vector](#)< Model >  
*finds all Models of the given key in the index.*
- auto [findManyId](#) (const Key &key) -> [Vector](#)< int >  
*finds all IDs of the given keys in the index.*

### 6.20.1 Detailed Description

```

template<typename Key, typename Model, typename DataFile>
class ticket::file::Index< Key, Model, DataFile >

```

Class representing an index file.

The [Index](#) maps Key to Model's numerical identifier, and provides methods to directly retrieve model objects from data files.

Model needs to be a subclass of [ManagedObject](#).

### 6.20.2 Constructor & Destructor Documentation

```

6.20.2.1 Index() template<typename Key , typename Model , typename DataFile >
ticket::file::Index< Key, Model, DataFile >::Index (
    Key Model::* ptr,
    const char * filename,
    DataFile & datafile )    [inline]

```

initializes the index.

## Parameters

<i>ptr</i>	the member pointer of the key.
<i>filename</i>	file to store the key.
<i>datafile</i>	the main file where data is stored.

## 6.20.3 Member Function Documentation

**6.20.3.1 findMany()** `template<typename Key , typename Model , typename DataFile >  
auto ticket::file::Index< Key, Model, DataFile >::findMany (`  
`const Key & key ) -> Vector<Model>     [inline]`

finds all Models of the given key in the index.

**6.20.3.2 findManyId()** `template<typename Key , typename Model , typename DataFile >  
auto ticket::file::Index< Key, Model, DataFile >::findManyId (`  
`const Key & key ) -> Vector<int>     [inline]`

finds all IDs of the given keys in the index.

**6.20.3.3 findOne()** `template<typename Key , typename Model , typename DataFile >  
auto ticket::file::Index< Key, Model, DataFile >::findOne (`  
`const Key & key ) -> Optional<Model>     [inline]`

finds one Model in the index.

**6.20.3.4 findOneId()** `template<typename Key , typename Model , typename DataFile >  
auto ticket::file::Index< Key, Model, DataFile >::findOneId (`  
`const Key & key ) -> Optional<int>     [inline]`

finds one identifier in the index.

**6.20.3.5 insert()** `template<typename Key , typename Model , typename DataFile >  
auto ticket::file::Index< Key, Model, DataFile >::insert (`  
`const Model & model ) -> void     [inline]`

inserts an object into the index.

```

6.20.3.6 remove() template<typename Key , typename Model , typename DataFile >
auto ticket::file::Index< Key, Model, DataFile >::remove (
    const Model & model ) -> void    [inline]

```

removes an object from the index.

The documentation for this class was generated from the following file:

- [lib/file/index.h](#)

## 6.21 ticket::file::Index< Varchar< maxLength >, Model, DataFile > Class Template Reference

Specialization of [Index](#) on [Varchar](#).

```
#include <index.h>
```

### Public Member Functions

- [Index](#) ([Key](#) Model::\*ptr, const char \*filename, DataFile &datafile)  
*initializes the index.*
- auto [insert](#) (const Model &model) -> void  
*inserts an object into the index.*
- auto [remove](#) (const Model &model) -> void  
*removes an object from the index.*
- auto [findOne](#) (const [Key](#) &key) -> [Optional](#)< Model >  
*finds one Model in the index.*
- auto [findOneId](#) (const [Key](#) &key) -> [Optional](#)< int >  
*finds one identifier in the index.*
- auto [findMany](#) (const [Key](#) &key) -> [Vector](#)< Model >  
*finds all Models of the given key in the index.*
- auto [findManyId](#) (const [Key](#) &key) -> [Vector](#)< int >  
*finds all IDs of the given keys in the index.*

### 6.21.1 Detailed Description

```

template<size_t maxLength, typename Model, typename DataFile>
class ticket::file::Index< Varchar< maxLength >, Model, DataFile >

```

Specialization of [Index](#) on [Varchar](#).

It makes use of hashes to speed up the process.

### 6.21.2 Constructor & Destructor Documentation

```

6.21.2.1 Index() template<size_t maxLength, typename Model , typename DataFile >
ticket::file::Index< Varchar< maxLength >, Model, DataFile >::Index (
    Key Model::* ptr,
    const char * filename,
    DataFile & datafile ) [inline]

```

initializes the index.

## Parameters

<i>ptr</i>	the member pointer of the key.
<i>filename</i>	file to store the key.
<i>datafile</i>	the main file where data is stored.

## 6.21.3 Member Function Documentation

**6.21.3.1 findMany()** `template<size_t maxLength, typename Model , typename DataFile >  
auto ticket::file::Index< Varchar< maxLength >, Model, DataFile >::findMany (  
    const Key & key ) -> Vector<Model>     [inline]`

finds all Models of the given key in the index.

**6.21.3.2 findManyId()** `template<size_t maxLength, typename Model , typename DataFile >  
auto ticket::file::Index< Varchar< maxLength >, Model, DataFile >::findManyId (  
    const Key & key ) -> Vector<int>     [inline]`

finds all IDs of the given keys in the index.

**6.21.3.3 findOne()** `template<size_t maxLength, typename Model , typename DataFile >  
auto ticket::file::Index< Varchar< maxLength >, Model, DataFile >::findOne (  
    const Key & key ) -> Optional<Model>     [inline]`

finds one Model in the index.

**6.21.3.4 findOneId()** `template<size_t maxLength, typename Model , typename DataFile >  
auto ticket::file::Index< Varchar< maxLength >, Model, DataFile >::findOneId (  
    const Key & key ) -> Optional<int>     [inline]`

finds one identifier in the index.

**6.21.3.5 insert()** `template<size_t maxLength, typename Model , typename DataFile >  
auto ticket::file::Index< Varchar< maxLength >, Model, DataFile >::insert (  
    const Model & model ) -> void     [inline]`

inserts an object into the index.



```
6.21.3.6 remove() template<size_t maxLength, typename Model , typename DataFile >
auto ticket::file::Index< Varchar< maxLength >, Model, DataFile >::remove (
    const Model & model ) -> void    [inline]
```

removes an object from the index.

The documentation for this class was generated from the following file:

- lib/file/[index.h](#)

## 6.22 ticket::Instant Class Reference

Class representing a point of time in a day.

```
#include <datetime.h>
```

### Public Member Functions

- [Instant](#) ()=default
- [Instant](#) (int [hour](#), int [minute](#))
- [Instant](#) (const char \*str)  
*constructs an [Instant](#) from an HH:MM format string.*
- auto [daysOverflow](#) () const -> int
- auto [hour](#) () const -> int
- auto [minute](#) () const -> int
- [operator std::string](#) () const  
*gets an HH:MM representation of the [Instant](#).*
- auto [operator+](#) ([Duration](#) dt) const -> [Instant](#)
- auto [operator-](#) ([Duration](#) dt) const -> [Instant](#)
- auto [operator-](#) ([Instant](#) rhs) const -> [Duration](#)
- auto [operator<](#) (const [Instant](#) &rhs) const -> bool

### 6.22.1 Detailed Description

Class representing a point of time in a day.

An [Instant](#) may overflow, and this class takes care of that by [daysOverflow\(\)](#).

Not to be confused with [Duration](#), see notes in [Duration](#).

### 6.22.2 Constructor & Destructor Documentation

**6.22.2.1 Instant()** [1/3] ticket::Instant::Instant ( ) [default]

**6.22.2.2 Instant()** [2/3] `ticket::Instant::Instant (`  
    `int hour,`  
    `int minute )`

**6.22.2.3 Instant()** [3/3] `ticket::Instant::Instant (`  
    `const char * str ) [explicit]`

constructs an [Instant](#) from an HH:MM format string.

### 6.22.3 Member Function Documentation

**6.22.3.1 daysOverflow()** `auto ticket::Instant::daysOverflow ( ) const -> int`

**6.22.3.2 hour()** `auto ticket::Instant::hour ( ) const -> int`

**6.22.3.3 minute()** `auto ticket::Instant::minute ( ) const -> int`

**6.22.3.4 operator std::string()** `ticket::Instant::operator std::string ( ) const`

gets an HH:MM representation of the [Instant](#).

**6.22.3.5 operator+()** `auto ticket::Instant::operator+ (`  
    `Duration dt ) const -> Instant`

**6.22.3.6 operator-()** [1/2] `auto ticket::Instant::operator- (`  
    `Duration dt ) const -> Instant`

**6.22.3.7 operator-()** [2/2] `auto ticket::Instant::operator- (`  
    `Instant rhs ) const -> Duration`

**6.22.3.8 operator<()** `auto ticket::Instant::operator< (`  
`const Instant & rhs ) const -> bool`

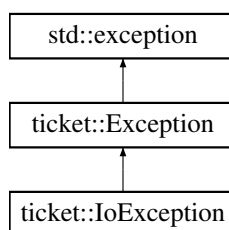
The documentation for this class was generated from the following file:

- [lib/datetime.h](#)

## 6.23 ticket::IoException Class Reference

```
#include <exception.h>
```

Inheritance diagram for ticket::IoException:



### Public Member Functions

- [IoException](#) ()
- [IoException](#) (const char \*[what](#))

### 6.23.1 Constructor & Destructor Documentation

**6.23.1.1 IoException()** [1/2] `ticket::IoException::IoException ( ) [inline]`

**6.23.1.2 IoException()** [2/2] `ticket::IoException::IoException (`  
`const char * what ) [inline]`

The documentation for this class was generated from the following file:

- [lib/exception.h](#)

## 6.24 ticket::HashMap< Key, Value, Hash, Equal >::iterator Class Reference

```
#include <hashmap.h>
```

## Public Types

- using `difference_type` = `std::ptrdiff_t`
- using `value_type` = `HashMap::value_type`
- using `pointer` = `value_type *`
- using `reference` = `value_type &`
- using `iterator_category` = `std::output_iterator_tag`

## Public Member Functions

- `iterator` ()=default
- `iterator` (ListNode \*node, HashMap \*home)
- auto `operator++` (int) -> `iterator`
- auto `operator++` () -> `iterator &`
- auto `operator--` (int) -> `iterator`
- auto `operator--` () -> `iterator &`
- auto `operator*` () const -> `reference`
- auto `operator==` (const `iterator` &rhs) const -> `bool`
- auto `operator==` (const `const_iterator` &rhs) const -> `bool`
- auto `operator!=` (const `iterator` &rhs) const -> `bool`
- auto `operator!=` (const `const_iterator` &rhs) const -> `bool`
- auto `operator->` () const noexcept -> `pointer`

## Friends

- class `const_iterator`
- class `HashMap`

### 6.24.1 Member Typedef Documentation

**6.24.1.1 `difference_type`** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`using ticket::HashMap< Key, Value, Hash, Equal >::iterator::difference_type = std::ptrdiff_t`

**6.24.1.2 `iterator_category`** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`using ticket::HashMap< Key, Value, Hash, Equal >::iterator::iterator_category = std::output_iterator_tag`

**6.24.1.3 `pointer`** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`using ticket::HashMap< Key, Value, Hash, Equal >::iterator::pointer = value_type *`

**6.24.1.4 reference** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::iterator::reference = value\_type &`

**6.24.1.5 value\_type** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
using ticket::HashMap< Key, Value, Hash, Equal >::iterator::value_type = HashMap::value\_type`

## 6.24.2 Constructor & Destructor Documentation

**6.24.2.1 iterator() [1/2]** `template<typename Key , typename Value , typename Hash = std::hash<↵  
Key>, typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::iterator::iterator ( ) [default]`

**6.24.2.2 iterator() [2/2]** `template<typename Key , typename Value , typename Hash = std::hash<↵  
Key>, typename Equal = std::equal_to<Key>>  
ticket::HashMap< Key, Value, Hash, Equal >::iterator::iterator (   
    ListNode * node,  
    HashMap * home ) [inline]`

## 6.24.3 Member Function Documentation

**6.24.3.1 operator"!="() [1/2]** `template<typename Key , typename Value , typename Hash = std::↵  
::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator!= (   
    const const\_iterator & rhs ) const -> bool [inline]`

**6.24.3.2 operator"!="() [2/2]** `template<typename Key , typename Value , typename Hash = std::↵  
::hash<Key>, typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator!= (   
    const iterator & rhs ) const -> bool [inline]`

**6.24.3.3 operator\*()** `template<typename Key , typename Value , typename Hash = std::hash<Key>,  
typename Equal = std::equal_to<Key>>  
auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator* ( ) const -> reference  
[inline]`

**6.24.3.4 operator++()** [1/2] `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator++ ( ) -> iterator & [inline]`

**6.24.3.5 operator++()** [2/2] `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator++ (`  
`int ) -> iterator [inline]`

**6.24.3.6 operator--()** [1/2] `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator-- ( ) -> iterator & [inline]`

**6.24.3.7 operator--()** [2/2] `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator-- (`  
`int ) -> iterator [inline]`

**6.24.3.8 operator->()** `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator-> ( ) const -> pointer [inline], [noexcept]`

**6.24.3.9 operator==()** [1/2] `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator== (`  
`const const_iterator & rhs ) const -> bool [inline]`

**6.24.3.10 operator==()** [2/2] `template<typename Key , typename Value , typename Hash = std::hash<Key>, typename Equal = std::equal_to<Key>>`  
`auto ticket::HashMap< Key, Value, Hash, Equal >::iterator::operator== (`  
`const iterator & rhs ) const -> bool [inline]`

## 6.24.4 Friends And Related Function Documentation

```
6.24.4.1 const_iterator template<typename Key , typename Value , typename Hash = std::hash<↵
Key>, typename Equal = std::equal_to<Key>>
friend class const_iterator [friend]
```

```
6.24.4.2 HashMap template<typename Key , typename Value , typename Hash = std::hash<Key>,
typename Equal = std::equal_to<Key>>
friend class HashMap [friend]
```

The documentation for this class was generated from the following file:

- lib/[hashmap.h](#)

## 6.25 ticket::Vector< T >::iterator Class Reference

```
#include <vector.h>
```

### Public Types

- using [difference\\_type](#) = std::ptrdiff\_t
- using [value\\_type](#) = T
- using [pointer](#) = T \*
- using [reference](#) = T &
- using [iterator\\_category](#) = std::output\_iterator\_tag

### Public Member Functions

- auto [operator+](#) (const int &n) const -> [iterator](#)
- auto [operator-](#) (const int &n) const -> [iterator](#)
- auto [operator-](#) (const [iterator](#) &rhs) const -> int
- auto [operator+=](#) (const int &n) -> [iterator](#) &
- auto [operator-=](#) (const int &n) -> [iterator](#) &
- auto [operator++](#) (int) const -> [iterator](#)
- auto [operator++](#) () -> [iterator](#) &
- auto [operator--](#) (int) const -> [iterator](#)
- auto [operator--](#) () -> [iterator](#) &
- auto [operator\\*](#) () const -> T &
- auto [operator==](#) (const [iterator](#) &rhs) const -> bool
- auto [operator==](#) (const [const\\_iterator](#) &rhs) const -> bool
- auto [operator!=](#) (const [iterator](#) &rhs) const -> bool
- auto [operator!=](#) (const [const\\_iterator](#) &rhs) const -> bool
- auto [operator<](#) (const [iterator](#) &rhs) const -> bool
- auto [operator<](#) (const [const\\_iterator](#) &rhs) const -> bool

### Friends

- class [const\\_iterator](#)
- class [Vector](#)

## 6.25.1 Member Typedef Documentation

**6.25.1.1 difference\_type** `template<typename T >`  
using `ticket::Vector< T >::iterator::difference_type` = `std::ptrdiff_t`

**6.25.1.2 iterator\_category** `template<typename T >`  
using `ticket::Vector< T >::iterator::iterator_category` = `std::output_iterator_tag`

**6.25.1.3 pointer** `template<typename T >`  
using `ticket::Vector< T >::iterator::pointer` = `T *`

**6.25.1.4 reference** `template<typename T >`  
using `ticket::Vector< T >::iterator::reference` = `T &`

**6.25.1.5 value\_type** `template<typename T >`  
using `ticket::Vector< T >::iterator::value_type` = `T`

## 6.25.2 Member Function Documentation

**6.25.2.1 operator!=( ) [1/2]** `template<typename T >`  
auto `ticket::Vector< T >::iterator::operator!= (`  
    const `const_iterator` & *rhs* ) const -> bool   [inline]

**6.25.2.2 operator!=( ) [2/2]** `template<typename T >`  
auto `ticket::Vector< T >::iterator::operator!= (`  
    const `iterator` & *rhs* ) const -> bool   [inline]

some other operator for iterator.

**6.25.2.3 operator\*( )** `template<typename T >`  
auto `ticket::Vector< T >::iterator::operator* ( )` const -> `T &`   [inline]



**6.25.2.4 operator+()** `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator+ (`  
`const int & n ) const -> iterator [inline]`

**6.25.2.5 operator++()** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator++ ( ) -> iterator & [inline]`

**6.25.2.6 operator++()** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator++ (`  
`int ) const -> iterator [inline]`

**6.25.2.7 operator+=()** `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator+= (`  
`const int & n ) -> iterator & [inline]`

**6.25.2.8 operator-()** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator- (`  
`const int & n ) const -> iterator [inline]`

**6.25.2.9 operator-()** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator- (`  
`const iterator & rhs ) const -> int [inline]`

**6.25.2.10 operator--()** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator-- ( ) -> iterator & [inline]`

**6.25.2.11 operator--()** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator-- (`  
`int ) const -> iterator [inline]`

**6.25.2.12 operator-=( )** `template<typename T >`  
`auto ticket::Vector< T >::iterator::operator-= (`  
`const int & n ) -> iterator &     [inline]`

**6.25.2.13 operator<( )** `[1/2] template<typename T >`  
`auto ticket::Vector< T >::iterator::operator< (`  
`const const_iterator & rhs ) const -> bool     [inline]`

**6.25.2.14 operator<( )** `[2/2] template<typename T >`  
`auto ticket::Vector< T >::iterator::operator< (`  
`const iterator & rhs ) const -> bool     [inline]`

**6.25.2.15 operator==( )** `[1/2] template<typename T >`  
`auto ticket::Vector< T >::iterator::operator==(`  
`const const_iterator & rhs ) const -> bool     [inline]`

**6.25.2.16 operator==( )** `[2/2] template<typename T >`  
`auto ticket::Vector< T >::iterator::operator==(`  
`const iterator & rhs ) const -> bool     [inline]`

a operator to check whether two iterators are same (pointing to the same memory address).

## 6.25.3 Friends And Related Function Documentation

**6.25.3.1 const\_iterator** `template<typename T >`  
`friend class const_iterator     [friend]`

**6.25.3.2 Vector** `template<typename T >`  
`friend class Vector     [friend]`

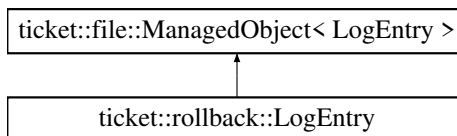
The documentation for this class was generated from the following file:

- [lib/vector.h](#)

## 6.26 ticket::rollback::LogEntry Struct Reference

```
#include <rollback.h>
```

Inheritance diagram for ticket::rollback::LogEntry:



### Public Attributes

- int [timestamp](#)
- [Variant](#)< [AddUser](#), [ModifyProfile](#), [AddTrain](#), [ReleaseTrain](#), [BuyTicket](#), [RefundTicket](#) > [content](#)

### Additional Inherited Members

#### 6.26.1 Member Data Documentation

**6.26.1.1** [content](#) [Variant](#)< [AddUser](#), [ModifyProfile](#), [AddTrain](#), [ReleaseTrain](#), [BuyTicket](#), [RefundTicket](#) > ticket::rollback::LogEntry::content

**6.26.1.2** [timestamp](#) int ticket::rollback::LogEntry::timestamp

The documentation for this struct was generated from the following file:

- [src/rollback.h](#)

## 6.27 ticket::command::Login Struct Reference

```
#include <parser.h>
```

### Public Attributes

- std::string [username](#)
- std::string [password](#)

#### 6.27.1 Member Data Documentation

**6.27.1.1 password** `std::string ticket::command::Login::password`

**6.27.1.2 username** `std::string ticket::command::Login::username`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.28 ticket::command::Logout Struct Reference

```
#include <parser.h>
```

### Public Attributes

- `std::string` [username](#)

### 6.28.1 Member Data Documentation

**6.28.1.1 username** `std::string ticket::command::Logout::username`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.29 ticket::file::ManagedObject< T, Meta, szChunk > Class Template Reference

an opinionated utility base class for the objects to be stored.

```
#include <file.h>
```

### Public Member Functions

- [ManagedObject](#) ([File\\_](#) &file)
- virtual [~ManagedObject](#) ()=default
- auto [id](#) () -> `size_t`  
*the unique immutable numeral identifier of the object.*
- auto [save](#) () -> void  
*saves the object into the file.*
- auto [update](#) () -> void  
*updates a modified object.*
- auto [destroy](#) () -> void  
*removes the object from the file.*

## Static Public Member Functions

- static auto `get (File_ &file, size_t id) -> T`  
*gets the object at id in file.*

### 6.29.1 Detailed Description

```
template<typename T, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
class ticket::file::ManagedObject< T, Meta, szChunk >
```

an opinionated utility base class for the objects to be stored.

it handles get, update, and push for the object.

### 6.29.2 Constructor & Destructor Documentation

```
6.29.2.1 ManagedObject() template<typename T , typename Meta = Unit, size_t szChunk = k↵
DefaultSzChunk>
ticket::file::ManagedObject< T, Meta, szChunk >::ManagedObject (
    File_ & file ) [inline]
```

```
6.29.2.2 ~ManagedObject() template<typename T , typename Meta = Unit, size_t szChunk = k↵
DefaultSzChunk>
virtual ticket::file::ManagedObject< T, Meta, szChunk >::~~ManagedObject ( ) [virtual], [default]
```

### 6.29.3 Member Function Documentation

```
6.29.3.1 destroy() template<typename T , typename Meta = Unit, size_t szChunk = kDefaultSz↵
Chunk>
auto ticket::file::ManagedObject< T, Meta, szChunk >::destroy ( ) -> void [inline]
```

removes the object from the file.

```
6.29.3.2 get() template<typename T , typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
static auto ticket::file::ManagedObject< T, Meta, szChunk >::get (
    File_ & file,
    size_t id ) -> T [inline], [static]
```

gets the object at id in file.

```
6.29.3.3 id() template<typename T , typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
auto ticket::file::ManagedObject< T, Meta, szChunk >::id ( ) -> size_t [inline]
```

the unique immutable numeral identifier of the object.

this identifier would not change on update, but may be reused when deleted.

```
6.29.3.4 save() template<typename T , typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
auto ticket::file::ManagedObject< T, Meta, szChunk >::save ( ) -> void [inline]
```

saves the object into the file.

The object needs to be new, i.e. not saved before. To update the object after a modification, use [update\(\)](#).

```
6.29.3.5 update() template<typename T , typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
auto ticket::file::ManagedObject< T, Meta, szChunk >::update ( ) -> void [inline]
```

updates a modified object.

The documentation for this class was generated from the following file:

- lib/file/[file.h](#)

## 6.30 ticket::Map< KeyType, ValueType, Compare > Class Template Reference

A sorted key-value map backed by a red-black tree.

```
#include <map.h>
```

### Public Types

- using [value\\_type](#) = [Pair](#)< const KeyType, ValueType >
- using [iterator](#) = typename TreeType::iterator
- using [const\\_iterator](#) = typename TreeType::const\_iterator

### Public Member Functions

- [Map](#) ()=default
- auto [at](#) (const KeyType &key) -> ValueType &
- auto [at](#) (const KeyType &key) const -> const ValueType &
- auto [operator\[\]](#) (const KeyType &key) -> ValueType &
- auto [operator\[\]](#) (const KeyType &key) const -> const ValueType &
- auto [begin](#) () -> [iterator](#)
- auto [cbegin](#) () const -> [const\\_iterator](#)
- auto [end](#) () -> [iterator](#)
- auto [cend](#) () const -> [const\\_iterator](#)
- auto [empty](#) () const -> bool
- auto [size](#) () const -> size\_t
- auto [clear](#) () -> void
- auto [insert](#) (const [value\\_type](#) &value) -> [Pair](#)< [iterator](#), bool >
- auto [erase](#) ([iterator](#) pos) -> void
- auto [count](#) (const KeyType &key) const -> size\_t
- auto [find](#) (const KeyType &key) -> [iterator](#)
- auto [find](#) (const KeyType &key) const -> [const\\_iterator](#)

### 6.30.1 Detailed Description

```
template<typename KeyType, typename ValueType, typename Compare = internal::LessOp>
class ticket::Map< KeyType, ValueType, Compare >
```

A sorted key-value map backed by a red-black tree.

### 6.30.2 Member Typedef Documentation

**6.30.2.1 const\_iterator** `template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>`  
`using ticket::Map< KeyType, ValueType, Compare >::const_iterator = typename TreeType::const_iterator`

**6.30.2.2 iterator** `template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>`  
`using ticket::Map< KeyType, ValueType, Compare >::iterator = typename TreeType::iterator`

**6.30.2.3 value\_type** `template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>`  
`using ticket::Map< KeyType, ValueType, Compare >::value_type = Pair<const KeyType, ValueType>`

### 6.30.3 Constructor & Destructor Documentation

**6.30.3.1 Map()** `template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>`  
`ticket::Map< KeyType, ValueType, Compare >::Map ( ) [default]`

### 6.30.4 Member Function Documentation

**6.30.4.1 at()** [1/2] `template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>`  
`auto ticket::Map< KeyType, ValueType, Compare >::at (`  
`const KeyType & key ) -> ValueType & [inline]`

access specified element with bounds checking Returns a reference to the mapped value of the element with key equivalent to key. If no such element exists, an exception of type 'index\_out\_of\_bound'

```
6.30.4.2 at() [2/2] template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::at (
    const KeyType & key ) const -> const ValueType &    [inline]
```

```
6.30.4.3 begin() template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::begin ( ) -> iterator    [inline]
```

return a iterator to the beginning

```
6.30.4.4 cbegin() template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::cbegin ( ) const -> const_iterator    [inline]
```

```
6.30.4.5 cend() template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::cend ( ) const -> const_iterator    [inline]
```

```
6.30.4.6 clear() template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::clear ( ) -> void    [inline]
```

clears the contents

```
6.30.4.7 count() template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::count (
    const KeyType & key ) const -> size_t    [inline]
```

Returns the number of elements with key that compares equivalent to the specified argument, which is either 1 or 0 since this container does not allow duplicates. The default method of check the equivalence is  $!(a < b \parallel b > a)$

```
6.30.4.8 empty() template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::empty ( ) const -> bool    [inline]
```

checks whether the container is empty return true if empty, otherwise false.

```
6.30.4.9 end() template<typename KeyType , typename ValueType , typename Compare = internal↵
::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::end ( ) -> iterator    [inline]
```

return a iterator to the end in fact, it returns past-the-end.



```
6.30.4.10 erase() template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::erase (
    iterator pos ) -> void    [inline]
```

erase the element at pos. throw if pos pointed to a bad element (pos == this->end() || pos points an element out of this)

```
6.30.4.11 find() [1/2] template<typename KeyType , typename ValueType , typename Compare =
internal::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::find (
    const KeyType & key ) -> iterator    [inline]
```

Finds an element with key equivalent to key. key value of the element to search for. Iterator to an element with key equivalent to key. If no such element is found, past-the-end (see end()) iterator is returned.

```
6.30.4.12 find() [2/2] template<typename KeyType , typename ValueType , typename Compare =
internal::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::find (
    const KeyType & key ) const -> const_iterator    [inline]
```

```
6.30.4.13 insert() template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::insert (
    const value_type & value ) -> Pair<iterator, bool>    [inline]
```

insert an element. return a pair, the first of the pair is the iterator to the new element (or the element that prevented the insertion), the second one is true if insert successfully, or false.

```
6.30.4.14 operator[]() [1/2] template<typename KeyType , typename ValueType , typename Compare =
internal::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::operator[] (
    const KeyType & key ) -> ValueType &    [inline]
```

access specified element Returns a reference to the value that is mapped to a key equivalent to key, performing an insertion if such key does not already exist.

```
6.30.4.15 operator[]() [2/2] template<typename KeyType , typename ValueType , typename Compare =
internal::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::operator[] (
    const KeyType & key ) const -> const ValueType &    [inline]
```

behave like at() throw index\_out\_of\_bound if such key does not exist.

```
6.30.4.16 size() template<typename KeyType , typename ValueType , typename Compare = internal::LessOp>
auto ticket::Map< KeyType, ValueType, Compare >::size ( ) const -> size_t    [inline]
```

returns the number of elements.

The documentation for this class was generated from the following file:

- lib/map.h

## 6.31 ticket::command::ModifyProfile Struct Reference

```
#include <parser.h>
```

### Public Attributes

- std::string [currentUser](#)
- std::string [username](#)
- [Optional](#)< std::string > [password](#)
- [Optional](#)< std::string > [name](#)
- [Optional](#)< std::string > [email](#)
- [Optional](#)< int > [privilege](#)

### 6.31.1 Member Data Documentation

**6.31.1.1 currentUser** std::string ticket::command::ModifyProfile::currentUser

**6.31.1.2 email** [Optional](#)<std::string> ticket::command::ModifyProfile::email

**6.31.1.3 name** [Optional](#)<std::string> ticket::command::ModifyProfile::name

**6.31.1.4 password** [Optional](#)<std::string> ticket::command::ModifyProfile::password

**6.31.1.5 privilege** [Optional](#)<int> ticket::command::ModifyProfile::privilege

**6.31.1.6 username** std::string ticket::command::ModifyProfile::username

The documentation for this struct was generated from the following file:

- src/[parser.h](#)

## 6.32 ticket::rollback::ModifyProfile Struct Reference

```
#include <rollback.h>
```

### Public Attributes

- int `id`
- `Optional< User::Password >` `password`
- `Optional< User::Name >` `name`
- `Optional< User::Email >` `email`
- `Optional< User::Privilege >` `privilege`

### 6.32.1 Member Data Documentation

**6.32.1.1 email** `Optional<User::Email>` `ticket::rollback::ModifyProfile::email`

**6.32.1.2 id** `int` `ticket::rollback::ModifyProfile::id`

**6.32.1.3 name** `Optional<User::Name>` `ticket::rollback::ModifyProfile::name`

**6.32.1.4 password** `Optional<User::Password>` `ticket::rollback::ModifyProfile::password`

**6.32.1.5 privilege** `Optional<User::Privilege>` `ticket::rollback::ModifyProfile::privilege`

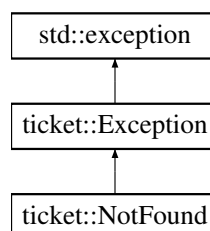
The documentation for this struct was generated from the following file:

- `src/rollback.h`

## 6.33 ticket::NotFound Class Reference

```
#include <exception.h>
```

Inheritance diagram for `ticket::NotFound`:



## Public Member Functions

- [NotFound](#) ()
- [NotFound](#) (const char \**what*)

### 6.33.1 Constructor & Destructor Documentation

**6.33.1.1 NotFound()** [1/2] `ticket::NotFound::NotFound ( ) [inline]`

**6.33.1.2 NotFound()** [2/2] `ticket::NotFound::NotFound ( const char * what ) [inline]`

The documentation for this class was generated from the following file:

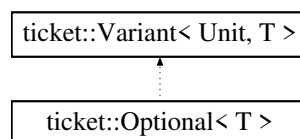
- [lib/exception.h](#)

## 6.34 ticket::Optional< T > Class Template Reference

A resemblance of std::optional.

```
#include <optional.h>
```

Inheritance diagram for ticket::Optional< T >:



## Public Member Functions

- [Optional](#) ()=default
- [Optional](#) (Unit)  
*constructs a empty optional.*
- [Optional](#) (const T &value)  
*constructs a filled optional.*
- auto [operator=](#) (const T &value) -> [Optional](#) &
- [operator bool](#) () const  
*true if the optional has value.*
- auto [operator\\*](#) () -> T &  
*provides access to the actual object.*
- auto [operator\\*](#) () const -> const T &
- auto [operator->](#) () -> T \*
- auto [operator->](#) () const -> const T \*

### 6.34.1 Detailed Description

```
template<typename T>
class ticket::Optional< T >
```

A resemblance of std::optional.

This class represents a state, or nothing at all. This is sometimes better than using null pointers, as it avoids the problem that a reference cannot be null. Internally it is a variant of `Unit` and `T`, therefore some may write `Optional<↵ T> = T? = T | Unit = T | null or whatever.`

### 6.34.2 Constructor & Destructor Documentation

**6.34.2.1 Optional()** [1/3] `template<typename T >`  
`ticket::Optional< T >::Optional ( ) [default]`

**6.34.2.2 Optional()** [2/3] `template<typename T >`  
`ticket::Optional< T >::Optional (`  
    `Unit ) [inline]`

constructs a empty optional.

**6.34.2.3 Optional()** [3/3] `template<typename T >`  
`ticket::Optional< T >::Optional (`  
    `const T & value ) [inline]`

constructs a filled optional.

### 6.34.3 Member Function Documentation

**6.34.3.1 operator bool()** `template<typename T >`  
`ticket::Optional< T >::operator bool ( ) const [inline]`

true if the optional has value.

**6.34.3.2 operator\*()** [1/2] `template<typename T >`  
`auto ticket::Optional< T >::operator* ( ) -> T & [inline]`

provides access to the actual object.

**6.34.3.3 operator\*()** [2/2] `template<typename T >`  
`auto ticket::Optional< T >::operator* ( ) const -> const T & [inline]`

**6.34.3.4 operator->()** [1/2] `template<typename T >`  
`auto ticket::Optional< T >::operator-> ( ) -> T * [inline]`

**6.34.3.5 operator->()** [2/2] `template<typename T >`  
`auto ticket::Optional< T >::operator-> ( ) const -> const T * [inline]`

**6.34.3.6 operator=()** `template<typename T >`  
`auto ticket::Optional< T >::operator= (`  
`const T & value ) -> Optional & [inline]`

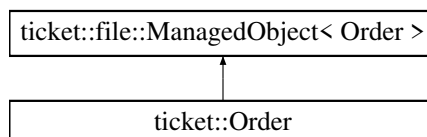
The documentation for this class was generated from the following file:

- [lib/optional.h](#)

## 6.35 ticket::Order Struct Reference

```
#include <order.h>
```

Inheritance diagram for ticket::Order:



### Public Types

- enum [Status](#) { [kSuccess](#) , [kPending](#) , [kRefunded](#) }
- using [Id](#) = int

## Public Member Functions

- auto `getTrain ()` -> `Train`  
*gets the corresponding train object.*

## Public Attributes

- `User::Id` `user`
- `Ride` `ride`
- int `ixFrom`
- int `ixTo`
- int `seats`
- `Status` `status`

## Additional Inherited Members

### 6.35.1 Member Typedef Documentation

**6.35.1.1** `Id` using `ticket::Order::Id` = int

### 6.35.2 Member Enumeration Documentation

**6.35.2.1** `Status` enum `ticket::Order::Status`

Enumerator

kSuccess	
kPending	
kRefunded	

### 6.35.3 Member Function Documentation

**6.35.3.1** `getTrain()` auto `ticket::Order::getTrain ( )` -> `Train`

gets the corresponding train object.

### 6.35.4 Member Data Documentation

**6.35.4.1 ixFrom** `int ticket::Order::ixFrom`

**6.35.4.2 ixTo** `int ticket::Order::ixTo`

**6.35.4.3 ride** `Ride ticket::Order::ride`

**6.35.4.4 seats** `int ticket::Order::seats`

**6.35.4.5 status** `Status ticket::Order::status`

**6.35.4.6 user** `User::Id ticket::Order::user`

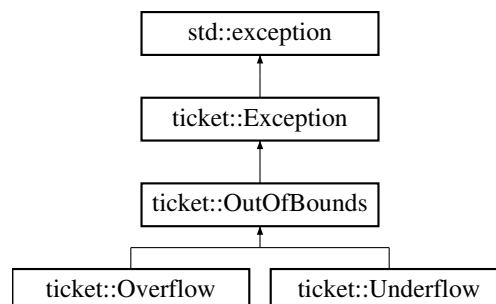
The documentation for this struct was generated from the following file:

- `src/order.h`

## 6.36 ticket::OutOfBounds Class Reference

```
#include <exception.h>
```

Inheritance diagram for ticket::OutOfBounds:



### Public Member Functions

- `OutOfBounds()`
- `OutOfBounds(const char *what)`



### 6.36.1 Constructor & Destructor Documentation

**6.36.1.1 OutOfBounds()** [1/2] `ticket::OutOfBounds::OutOfBounds ( ) [inline]`

**6.36.1.2 OutOfBounds()** [2/2] `ticket::OutOfBounds::OutOfBounds ( const char * what ) [inline]`

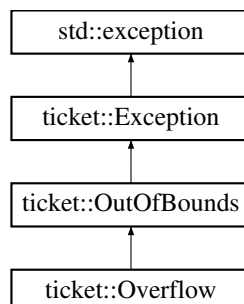
The documentation for this class was generated from the following file:

- [lib/exception.h](#)

## 6.37 ticket::Overflow Class Reference

```
#include <exception.h>
```

Inheritance diagram for ticket::Overflow:



### Public Member Functions

- [Overflow](#) ( )
- [Overflow](#) (const char \*[what](#))

### 6.37.1 Constructor & Destructor Documentation

**6.37.1.1 Overflow()** [1/2] `ticket::Overflow::Overflow ( ) [inline]`

**6.37.1.2 Overflow()** [2/2] `ticket::Overflow::Overflow (`  
`const char * what ) [inline]`

The documentation for this class was generated from the following file:

- [lib/exception.h](#)

## 6.38 ticket::Pair< T1, T2 > Class Template Reference

A pair of objects.

```
#include <utility.h>
```

### Public Member Functions

- constexpr [Pair](#) ()
- [Pair](#) (const [Pair](#) &other)=default
- [Pair](#) ([Pair](#) &&other) noexcept=default
- [Pair](#) (const T1 &x, const T2 &y)
- template<class U1 , class U2 >  
  [Pair](#) (U1 &&x, U2 &&y)
- template<class U1 , class U2 >  
  [Pair](#) (const [Pair](#)< U1, U2 > &other)
- template<class U1 , class U2 >  
  [Pair](#) ([Pair](#)< U1, U2 > &&other)

### Public Attributes

- T1 [first](#)
- T2 [second](#)

### 6.38.1 Detailed Description

```
template<typename T1, typename T2>  
class ticket::Pair< T1, T2 >
```

A pair of objects.

### 6.38.2 Constructor & Destructor Documentation

**6.38.2.1 Pair()** [1/7] `template<typename T1 , typename T2 >`  
`constexpr ticket::Pair< T1, T2 >::Pair ( ) [inline], [constexpr]`

**6.38.2.2 Pair()** [2/7] `template<typename T1 , typename T2 >`  
`ticket::Pair< T1, T2 >::Pair (`  
`const Pair< T1, T2 > & other )` [default]

**6.38.2.3 Pair()** [3/7] `template<typename T1 , typename T2 >`  
`ticket::Pair< T1, T2 >::Pair (`  
`Pair< T1, T2 > && other )` [default], [noexcept]

**6.38.2.4 Pair()** [4/7] `template<typename T1 , typename T2 >`  
`ticket::Pair< T1, T2 >::Pair (`  
`const T1 & x,`  
`const T2 & y )` [inline]

**6.38.2.5 Pair()** [5/7] `template<typename T1 , typename T2 >`  
`template<class U1 , class U2 >`  
`ticket::Pair< T1, T2 >::Pair (`  
`U1 && x,`  
`U2 && y )` [inline]

**6.38.2.6 Pair()** [6/7] `template<typename T1 , typename T2 >`  
`template<class U1 , class U2 >`  
`ticket::Pair< T1, T2 >::Pair (`  
`const Pair< U1, U2 > & other )` [inline]

**6.38.2.7 Pair()** [7/7] `template<typename T1 , typename T2 >`  
`template<class U1 , class U2 >`  
`ticket::Pair< T1, T2 >::Pair (`  
`Pair< U1, U2 > && other )` [inline]

## 6.38.3 Member Data Documentation

**6.38.3.1 first** `template<typename T1 , typename T2 >`  
`T1 ticket::Pair< T1, T2 >::first`

**6.38.3.2 second** `template<typename T1 , typename T2 >`  
`T2 ticket::Pair< T1, T2 >::second`

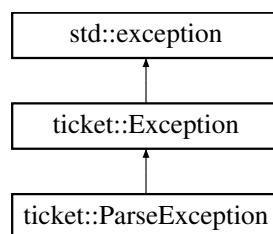
The documentation for this class was generated from the following file:

- [lib/utility.h](#)

## 6.39 ticket::ParseException Class Reference

```
#include <exception.h>
```

Inheritance diagram for ticket::ParseException:



### Public Member Functions

- [ParseException\(\)](#)
- [ParseException](#) (const char \**what*)

### 6.39.1 Constructor & Destructor Documentation

**6.39.1.1 ParseException()** [1/2] `ticket::ParseException::ParseException ( ) [inline]`

**6.39.1.2 ParseException()** [2/2] `ticket::ParseException::ParseException (`  
`const char * what ) [inline]`

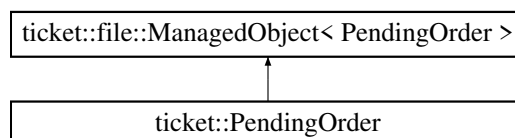
The documentation for this class was generated from the following file:

- [lib/exception.h](#)

## 6.40 ticket::PendingOrder Struct Reference

```
#include <order.h>
```

Inheritance diagram for ticket::PendingOrder:



## Public Member Functions

- auto `satisfiable` () -> bool  
*checks if the order is satisfiable.*
- auto `getOrder` () -> `Order`  
*gets the corresponding order object.*

## Public Attributes

- `Ride ride`
- int `ixFrom`
- int `ixTo`
- int `seats`
- `Order::Id order`

## Additional Inherited Members

### 6.40.1 Member Function Documentation

**6.40.1.1 `getOrder()`** auto `ticket::PendingOrder::getOrder` ( ) -> `Order`

gets the corresponding order object.

**6.40.1.2 `satisfiable()`** auto `ticket::PendingOrder::satisfiable` ( ) -> bool

checks if the order is satisfiable.

### 6.40.2 Member Data Documentation

**6.40.2.1 `ixFrom`** int `ticket::PendingOrder::ixFrom`

**6.40.2.2 `ixTo`** int `ticket::PendingOrder::ixTo`

**6.40.2.3 `order`** `Order::Id` `ticket::PendingOrder::order`

**6.40.2.4 ride** `Ride ticket::PendingOrder::ride`

**6.40.2.5 seats** `int ticket::PendingOrder::seats`

The documentation for this struct was generated from the following file:

- `src/order.h`

## 6.41 ticket::command::QueryOrder Struct Reference

```
#include <parser.h>
```

### Public Attributes

- `std::string currentUser`

### 6.41.1 Member Data Documentation

**6.41.1.1 currentUser** `std::string ticket::command::QueryOrder::currentUser`

The documentation for this struct was generated from the following file:

- `src/parser.h`

## 6.42 ticket::command::QueryProfile Struct Reference

```
#include <parser.h>
```

### Public Attributes

- `std::string currentUser`
- `std::string username`

### 6.42.1 Member Data Documentation

**6.42.1.1 currentUser** `std::string ticket::command::QueryProfile::currentUser`

**6.42.1.2 username** `std::string ticket::command::QueryProfile::username`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.43 ticket::command::QueryTicket Struct Reference

```
#include <parser.h>
```

### Public Attributes

- `std::string` [from](#)
- `std::string` [to](#)
- [Date](#) [date](#)
- [SortType](#) [sort](#) = [kTime](#)

### 6.43.1 Member Data Documentation

**6.43.1.1 date** [Date](#) `ticket::command::QueryTicket::date`

**6.43.1.2 from** `std::string ticket::command::QueryTicket::from`

**6.43.1.3 sort** [SortType](#) `ticket::command::QueryTicket::sort = kTime`

**6.43.1.4 to** `std::string ticket::command::QueryTicket::to`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.44 ticket::command::QueryTrain Struct Reference

```
#include <parser.h>
```

## Public Attributes

- std::string [id](#)
- [Date](#) [date](#)

### 6.44.1 Member Data Documentation

**6.44.1.1** **date** [Date](#) ticket::command::QueryTrain::date

**6.44.1.2** **id** std::string ticket::command::QueryTrain::id

The documentation for this struct was generated from the following file:

- src/[parser.h](#)

## 6.45 ticket::command::QueryTransfer Struct Reference

```
#include <parser.h>
```

## Public Attributes

- std::string [from](#)
- std::string [to](#)
- [Date](#) [date](#)
- [SortType](#) [sort](#) = [kTime](#)

### 6.45.1 Member Data Documentation

**6.45.1.1** **date** [Date](#) ticket::command::QueryTransfer::date

**6.45.1.2** **from** std::string ticket::command::QueryTransfer::from

**6.45.1.3** **sort** [SortType](#) ticket::command::QueryTransfer::sort = [kTime](#)



**6.45.1.4** `to` `std::string ticket::command::QueryTransfer::to`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.46 `ticket::command::RefundTicket` Struct Reference

```
#include <parser.h>
```

### Public Attributes

- `std::string` [currentUser](#)
- `int` [index](#) = 1

#### 6.46.1 Member Data Documentation

**6.46.1.1** `currentUser` `std::string ticket::command::RefundTicket::currentUser`

**6.46.1.2** `index` `int ticket::command::RefundTicket::index = 1`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.47 `ticket::rollback::RefundTicket` Struct Reference

```
#include <rollback.h>
```

### Public Attributes

- `int` [id](#)
- [Order::Status](#) [status](#)

#### 6.47.1 Member Data Documentation

**6.47.1.1 id** `int ticket::rollback::RefundTicket::id`

**6.47.1.2 status** `Order::Status ticket::rollback::RefundTicket::status`

The documentation for this struct was generated from the following file:

- [src/rollback.h](#)

## 6.48 ticket::command::ReleaseTrain Struct Reference

```
#include <parser.h>
```

### Public Attributes

- `std::string id`

#### 6.48.1 Member Data Documentation

**6.48.1.1 id** `std::string ticket::command::ReleaseTrain::id`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.49 ticket::rollback::ReleaseTrain Struct Reference

```
#include <rollback.h>
```

### Public Attributes

- `int id`

#### 6.49.1 Member Data Documentation

**6.49.1.1 id** `int ticket::rollback::ReleaseTrain::id`

The documentation for this struct was generated from the following file:

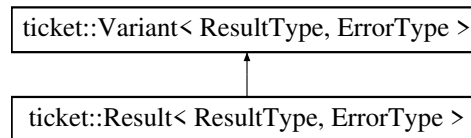
- [src/rollback.h](#)

## 6.50 ticket::Result< ResultType, ErrorType > Class Template Reference

Result<Res, Err> = Res | Err.

```
#include <result.h>
```

Inheritance diagram for ticket::Result< ResultType, ErrorType >:



### Public Member Functions

- [Result](#) ()=delete
  - template<typename T >  
  [Result](#) (const T &value)
  - auto [result](#) () -> ResultType &
  - auto [result](#) () const -> const ResultType &
  - auto [error](#) () -> ErrorType \*
  - auto [error](#) () const -> const ErrorType \*
  - auto [success](#) () const -> bool
- returns true if the result is in its successful state.*

### 6.50.1 Detailed Description

```
template<typename ResultType, typename ErrorType>
class ticket::Result< ResultType, ErrorType >
```

Result<Res, Err> = Res | Err.

This class provides a wrapper around variant to make error handling a little easier. Recommended usage:

```
auto foo = doSomethingThatMightFail(args);
if (auto err = foo.error()) {
    // handles error, or rethrow:
    return *err;
}
std::cout << foo.result() << std::endl;
```

Therefore, [result\(\)](#) returns a reference, while [error\(\)](#) returns a pointer. This design is subject to change.

### 6.50.2 Constructor & Destructor Documentation

```
6.50.2.1 Result() [1/2] template<typename ResultType , typename ErrorType >
ticket::Result< ResultType, ErrorType >::Result ( ) [delete]
```

```
6.50.2.2 Result() [2/2] template<typename ResultType , typename ErrorType >
template<typename T >
ticket::Result< ResultType, ErrorType >::Result (
    const T & value ) [inline]
```

### 6.50.3 Member Function Documentation

```
6.50.3.1 error() [1/2] template<typename ResultType , typename ErrorType >
auto ticket::Result< ResultType, ErrorType >::error ( ) -> ErrorType * [inline]
```

```
6.50.3.2 error() [2/2] template<typename ResultType , typename ErrorType >
auto ticket::Result< ResultType, ErrorType >::error ( ) const -> const ErrorType * [inline]
```

```
6.50.3.3 result() [1/2] template<typename ResultType , typename ErrorType >
auto ticket::Result< ResultType, ErrorType >::result ( ) -> ResultType & [inline]
```

```
6.50.3.4 result() [2/2] template<typename ResultType , typename ErrorType >
auto ticket::Result< ResultType, ErrorType >::result ( ) const -> const ResultType & [inline]
```

```
6.50.3.5 success() template<typename ResultType , typename ErrorType >
auto ticket::Result< ResultType, ErrorType >::success ( ) const -> bool [inline]
```

returns true if the result is in its successful state.

The documentation for this class was generated from the following file:

- [lib/result.h](#)

## 6.51 ticket::Ride Struct Reference

```
#include <train.h>
```

### Public Member Functions

- auto [operator](#)< (const [Ride](#) &rhs) const -> bool

## Public Attributes

- int [train](#)  
*the numerical id of the train.*
- [Date](#) [date](#)

### 6.51.1 Member Function Documentation

**6.51.1.1** `operator<()` `auto ticket::Ride::operator< (`  
`const Ride & rhs ) const -> bool`

### 6.51.2 Member Data Documentation

**6.51.2.1** `date` `Date ticket::Ride::date`

**6.51.2.2** `train` `int ticket::Ride::train`

the numerical id of the train.

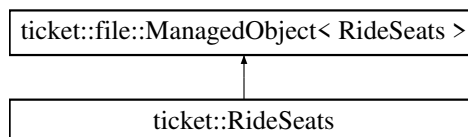
The documentation for this struct was generated from the following file:

- src/[train.h](#)

## 6.52 ticket::RideSeats Struct Reference

```
#include <train.h>
```

Inheritance diagram for ticket::RideSeats:



## Public Member Functions

- auto [ticketsAvailable](#) (int ixFrom, int ixTo) -> int  
*calculates how many tickets are still available.*

## Public Attributes

- [Ride ride](#)
- [file::Array< int, 99 > seatsRemaining](#)

## Additional Inherited Members

### 6.52.1 Member Function Documentation

**6.52.1.1 ticketsAvailable()** `auto ticket::RideSeats::ticketsAvailable (`  
     `int ixFrom,`  
     `int ixTo ) -> int`

calculates how many tickets are still available.

#### Parameters

<i>ixFrom</i>	index of the departing stop
<i>ixTo</i>	index of the arriving stop

### 6.52.2 Member Data Documentation

**6.52.2.1 ride** [Ride](#) `ticket::RideSeats::ride`

**6.52.2.2 seatsRemaining** [file::Array<int, 99>](#) `ticket::RideSeats::seatsRemaining`

The documentation for this struct was generated from the following file:

- [src/train.h](#)

## 6.53 ticket::command::Rollback Struct Reference

```
#include <parser.h>
```

## Public Attributes

- `int` [timestamp](#)

### 6.53.1 Member Data Documentation

#### 6.53.1.1 timestamp `int ticket::command::Rollback::timestamp`

The documentation for this struct was generated from the following file:

- [src/parser.h](#)

## 6.54 `ticket::file::Set< T, maxLength, Cmp >` Struct Template Reference

A sorted array with utility functions and bound checks.

```
#include <set.h>
```

### Public Member Functions

- `Set` ()=default
- auto `indexOfInsert` (const T &element) -> size\_t
- auto `indexOf` (const T &element) -> size\_t  
*finds the index of element in the set.*
- auto `includes` (const T &element) -> bool  
*checks if the elements is included in the set.*
- auto `insert` (const T &element) -> void  
*inserts the element into the set.*
- auto `remove` (const T &element) -> void  
*removes the element from the set.*
- auto `removeAt` (size\_t offset) -> void  
*removes the element at offset.*
- auto `clear` () -> void  
*clears the set.*
- void `copyFrom` (const `Set` &other, size\_t ixFrom, size\_t ixTo, size\_t count)  
*copies a portion of another set to this.*
- auto `operator[]` (size\_t index) -> T &
- auto `operator[]` (size\_t index) const -> const T &
- auto `pop` () -> T  
*pops the greatest element.*
- auto `shift` () -> T  
*pops the least element.*
- template<typename Functor >  
auto `forEach` (const Functor &callback) -> void  
*calls the callback for each element in the array.*

### Public Attributes

- size\_t `length` = 0
- T `content` [maxLength]

### 6.54.1 Detailed Description

```
template<typename T, size_t maxLength, typename Cmp = Less<>>
struct ticket::file::Set< T, maxLength, Cmp >
```

A sorted array with utility functions and bound checks.

### 6.54.2 Constructor & Destructor Documentation

**6.54.2.1 Set()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`ticket::file::Set< T, maxLength, Cmp >::Set ( ) [default]`

### 6.54.3 Member Function Documentation

**6.54.3.1 clear()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::clear ( ) -> void [inline]`

clears the set.

**6.54.3.2 copyFrom()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`void ticket::file::Set< T, maxLength, Cmp >::copyFrom (`  
`const Set< T, maxLength, Cmp > & other,`  
`size_t ixFrom,`  
`size_t ixTo,`  
`size_t count ) [inline]`

copies a portion of another set to this.

**6.54.3.3 forEach()** `template<typename T , size_t maxLength, typename Cmp = Less<>>`  
`template<typename Functor >`  
`auto ticket::file::Set< T, maxLength, Cmp >::forEach (`  
`const Functor & callback ) -> void [inline]`

calls the callback for each element in the array.



**6.54.3.4 includes()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::includes (`  
`const T & element ) -> bool     [inline]`

checks if the elements is included in the set.

**6.54.3.5 indexOf()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::indexOf (`  
`const T & element ) -> size_t     [inline]`

finds the index of element in the set.

**6.54.3.6 indexOfInsert()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::indexOfInsert (`  
`const T & element ) -> size_t     [inline]`

**6.54.3.7 insert()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::insert (`  
`const T & element ) -> void     [inline]`

inserts the element into the set.

**6.54.3.8 operator[]() [1/2]** `template<typename T , size_t maxLength, typename Cmp = Less<>>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::operator[] (`  
`size_t index ) -> T &     [inline]`

**6.54.3.9 operator[]() [2/2]** `template<typename T , size_t maxLength, typename Cmp = Less<>>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::operator[] (`  
`size_t index ) const -> const T &     [inline]`

**6.54.3.10 pop()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>`  
`auto ticket::file::Set< T, maxLength, Cmp >::pop ( ) -> T     [inline]`

pops the greatest element.

**6.54.3.11 remove()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>  
auto ticket::file::Set< T, maxLength, Cmp >::remove (   
 const T & element ) -> void [inline]`

removes the element from the set.

**6.54.3.12 removeAt()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>  
auto ticket::file::Set< T, maxLength, Cmp >::removeAt (   
 size_t offset ) -> void [inline]`

removes the element at offset.

**6.54.3.13 shift()** `template<typename T , size_t maxLength, typename Cmp = Less<>>>  
auto ticket::file::Set< T, maxLength, Cmp >::shift ( ) -> T [inline]`

pops the least element.

## 6.54.4 Member Data Documentation

**6.54.4.1 content** `template<typename T , size_t maxLength, typename Cmp = Less<>>>  
T ticket::file::Set< T, maxLength, Cmp >::content[maxLength]`

**6.54.4.2 length** `template<typename T , size_t maxLength, typename Cmp = Less<>>>  
size_t ticket::file::Set< T, maxLength, Cmp >::length = 0`

The documentation for this struct was generated from the following file:

- [lib/file/set.h](#)

## 6.55 ticket::Train::Stop Struct Reference

```
#include <train.h>
```

### Public Attributes

- [Station::Id name](#)

## 6.55.1 Member Data Documentation

### 6.55.1.1 name `Station::Id` `ticket::Train::Stop::name`

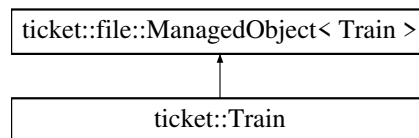
The documentation for this struct was generated from the following file:

- `src/train.h`

## 6.56 ticket::Train Struct Reference

```
#include <train.h>
```

Inheritance diagram for `ticket::Train`:



### Classes

- struct `Edge`
- struct `Stop`

### Public Types

- using `Id` = `file::Varchar< 20 >`
- using `Type` = `char`

### Public Member Functions

- auto `indexOfStop` (const std::string &name) -> `Result< int, NotFound >`  
*finds the index of the station of the given name.*
- auto `totalPrice` (int ixDeparture, int ixArrival) -> `int`  
*calculates the total price of a trip.*
- auto `getRide` (`Date` date) -> `RideSeats`  
*gets the remaining seats object on a given date.*
- auto `getRide` (`Date` date, int ixDeparture) -> `RideSeats`  
*gets the remaining seats object on a given date at a given stop.*
- auto `runsOnDate` (`Date` date) -> `bool`  
*checks if the train has a ride departing from the first station on the given date.*
- auto `runsOnDate` (`Date` date, int ixDeparture) -> `bool`  
*checks if the train has a ride departing from the given station on the given date.*

## Public Attributes

- [Id](#) [trainId](#)
- [file::Array](#)< [Stop](#), 100 > [stops](#)
- [file::Array](#)< [Edge](#), 99 > [edges](#)
- [int](#) [seats](#)
- [Date](#) [begin](#)
- [Date](#) [end](#)
- [Type](#) [type](#)
- [bool](#) [released](#)

## Additional Inherited Members

### 6.56.1 Member Typedef Documentation

**6.56.1.1 Id** using [ticket::Train::Id](#) = [file::Varchar](#)<20>

**6.56.1.2 Type** using [ticket::Train::Type](#) = [char](#)

### 6.56.2 Member Function Documentation

**6.56.2.1 getRide()** [1/2] auto [ticket::Train::getRide](#) (  
[Date](#) [date](#) ) -> [RideSeats](#)

gets the remaining seats object on a given date.

#### Parameters

<a href="#">date</a>	the departure date of the entire train (i.e. not the departure date of a stop).
----------------------	---

**6.56.2.2 getRide()** [2/2] auto [ticket::Train::getRide](#) (  
[Date](#) [date](#),  
[int](#) [ixDeparture](#) ) -> [RideSeats](#)

gets the remaining seats object on a given date at a given stop.

#### Parameters

<a href="#">date</a>	the departure date of a stop.
<a href="#">ixDeparture</a>	the index of the departing stop.

**6.56.2.3 indexOfStop()** `auto ticket::Train::indexOfStop (`  
`const std::string & name ) -> Result< int, NotFound >`

finds the index of the station of the given name.

**6.56.2.4 runsOnDate()** [1/2] `auto ticket::Train::runsOnDate (`  
`Date date ) -> bool`

checks if the train has a ride departing from the first station on the given date.

**Parameters**

<i>date</i>	the departure date of the first station.
-------------	--

**6.56.2.5 runsOnDate()** [2/2] `auto ticket::Train::runsOnDate (`  
`Date date,`  
`int ixDeparture ) -> bool`

checks if the train has a ride departing from the given station on the given date.

**Parameters**

<i>date</i>	the departure date of the given station.
<i>ixDeparture</i>	the index of the departing stop.

**6.56.2.6 totalPrice()** `auto ticket::Train::totalPrice (`  
`int ixDeparture,`  
`int ixArrival ) -> int`

calculates the total price of a trip.

## 6.56.3 Member Data Documentation

**6.56.3.1 begin** `Date ticket::Train::begin`

**6.56.3.2 edges** `file::Array<Edge, 99> ticket::Train::edges`

**6.56.3.3 end** `Date ticket::Train::end`

**6.56.3.4 released** `bool ticket::Train::released`

**6.56.3.5 seats** `int ticket::Train::seats`

**6.56.3.6 stops** `file::Array<Stop, 100> ticket::Train::stops`

**6.56.3.7 trainId** `Id ticket::Train::trainId`

**6.56.3.8 type** `Type ticket::Train::type`

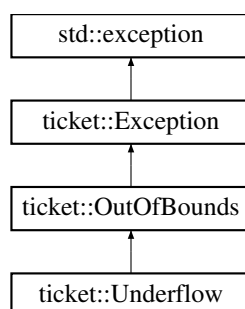
The documentation for this struct was generated from the following file:

- [src/train.h](#)

## 6.57 ticket::Underflow Class Reference

```
#include <exception.h>
```

Inheritance diagram for ticket::Underflow:



## Public Member Functions

- [Underflow](#) ()
- [Underflow](#) (const char \*[what](#))

### 6.57.1 Constructor & Destructor Documentation

**6.57.1.1 Underflow()** [1/2] `ticket::Underflow::Underflow ( ) [inline]`

**6.57.1.2 Underflow()** [2/2] `ticket::Underflow::Underflow ( const char * what ) [inline]`

The documentation for this class was generated from the following file:

- lib/[exception.h](#)

## 6.58 ticket::Unit Struct Reference

An empty class, used at various places.

```
#include <utility.h>
```

## Public Member Functions

- constexpr [Unit](#) ()=default
- template<typename T >  
constexpr [Unit](#) (const T &)
- auto [operator](#)< (const [Unit](#) &) -> bool

### 6.58.1 Detailed Description

An empty class, used at various places.

### 6.58.2 Constructor & Destructor Documentation

**6.58.2.1 Unit()** [1/2] `constexpr ticket::Unit::Unit ( ) [constexpr], [default]`

```

6.58.2.2 Unit() [2/2] template<typename T >
constexpr ticket::Unit::Unit (
    const T & ) [inline], [constexpr]

```

### 6.58.3 Member Function Documentation

```

6.58.3.1 operator<() auto ticket::Unit::operator< (
    const Unit & ) -> bool [inline]

```

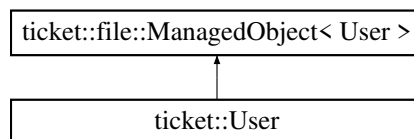
The documentation for this struct was generated from the following file:

- lib/utility.h

## 6.59 ticket::User Struct Reference

```
#include <user.h>
```

Inheritance diagram for ticket::User:



### Public Types

- using `Id` = `file::Varchar< 20 >`
- using `Password` = `file::Varchar< 30 >`
- using `Name` = `file::Varchar< 15 >`
- using `Email` = `file::Varchar< 30 >`
- using `Privilege` = `int`

### Static Public Member Functions

- static auto `hasUser` (`const char *username`) -> `bool`  
*checks if there is a user with the given username.*

### Public Attributes

- `Id username`
- `Password password`
- `Name name`
- `Email email`
- `Privilege privilege`



## Additional Inherited Members

### 6.59.1 Member Typedef Documentation

**6.59.1.1 Email** using `ticket::User::Email = file::Varchar<30>`

**6.59.1.2 Id** using `ticket::User::Id = file::Varchar<20>`

**6.59.1.3 Name** using `ticket::User::Name = file::Varchar<15>`

**6.59.1.4 Password** using `ticket::User::Password = file::Varchar<30>`

**6.59.1.5 Privilege** using `ticket::User::Privilege = int`

### 6.59.2 Member Function Documentation

**6.59.2.1 hasUser()** auto `ticket::User::hasUser (`  
    `const char * username ) -> bool [static]`

checks if there is a user with the given username.

### 6.59.3 Member Data Documentation

**6.59.3.1 email** `Email ticket::User::email`

**6.59.3.2 name** `Name ticket::User::name`

**6.59.3.3 password** `Password` ticket::User::password

**6.59.3.4 privilege** `Privilege` ticket::User::privilege

**6.59.3.5 username** `Id` ticket::User::username

The documentation for this struct was generated from the following files:

- src/user.h
- src/user.cpp

## 6.60 ticket::file::Varchar< maxLength > Struct Template Reference

A wrapper for const char \* with utility functions and type conversions.

```
#include <varchar.h>
```

### Public Member Functions

- `Varchar` ()
- `Varchar` (const std::string &s)
- `Varchar` (const char \*cstr)
- template<int A>  
  `Varchar` (const `Varchar`< A > &that)
- `operator std::string` () const
- auto `str` () const -> std::string
- auto `length` () -> int
- template<int A>  
  auto `operator=` (const `Varchar`< A > &that) -> `Varchar` &
- template<int A>  
  auto `operator<` (const `Varchar`< A > &that) const -> bool
- template<int A>  
  auto `operator==` (const `Varchar`< A > &that) const -> bool
- template<int A>  
  auto `operator!=` (const `Varchar`< A > &that) const -> bool
- auto `hash` () const -> size\_t

### Friends

- template<int A>  
  class `Varchar`

### 6.60.1 Detailed Description

```
template<int maxLength>
struct ticket::file::Varchar< maxLength >
```

A wrapper for `const char *` with utility functions and type conversions.

the trailing zero is not counted in `maxLength`.

its default ordering is hash order. this is for a maximum performance. you need to write a comparator if you want dictionary order.

### 6.60.2 Constructor & Destructor Documentation

**6.60.2.1 Varchar()** [1/4] `template<int maxLength>`  
`ticket::file::Varchar< maxLength >::Varchar ( )` [inline]

**6.60.2.2 Varchar()** [2/4] `template<int maxLength>`  
`ticket::file::Varchar< maxLength >::Varchar (`  
`const std::string & s )` [inline]

**6.60.2.3 Varchar()** [3/4] `template<int maxLength>`  
`ticket::file::Varchar< maxLength >::Varchar (`  
`const char * cstr )` [inline]

**6.60.2.4 Varchar()** [4/4] `template<int maxLength>`  
`template<int A>`  
`ticket::file::Varchar< maxLength >::Varchar (`  
`const Varchar< A > & that )` [inline]

### 6.60.3 Member Function Documentation

**6.60.3.1 hash()** `template<int maxLength>`  
`auto ticket::file::Varchar< maxLength >::hash ( ) const -> size_t` [inline]

**6.60.3.2 length()** template<int maxLength>

```
auto ticket::file::Varchar< maxLength >::length ( ) -> int    [inline]
```

**6.60.3.3 operator std::string()** template<int maxLength>

```
ticket::file::Varchar< maxLength >::operator std::string ( ) const    [inline]
```

**6.60.3.4 operator"!="()** template<int maxLength>

```
template<int A>
```

```
auto ticket::file::Varchar< maxLength >::operator!= (
    const Varchar< A > & that ) const -> bool    [inline]
```

**6.60.3.5 operator<()** template<int maxLength>

```
template<int A>
```

```
auto ticket::file::Varchar< maxLength >::operator< (
    const Varchar< A > & that ) const -> bool    [inline]
```

**6.60.3.6 operator=()** template<int maxLength>

```
template<int A>
```

```
auto ticket::file::Varchar< maxLength >::operator= (
    const Varchar< A > & that ) -> Varchar &    [inline]
```

**6.60.3.7 operator==()** template<int maxLength>

```
template<int A>
```

```
auto ticket::file::Varchar< maxLength >::operator== (
    const Varchar< A > & that ) const -> bool    [inline]
```

**6.60.3.8 str()** template<int maxLength>

```
auto ticket::file::Varchar< maxLength >::str ( ) const -> std::string    [inline]
```

**6.60.4 Friends And Related Function Documentation**

```

6.60.4.1 Varchar  template<int maxLength>
template<int A>
friend class Varchar [friend]

```

The documentation for this struct was generated from the following file:

- lib/file/[varchar.h](#)

## 6.61 ticket::Variant< Ts > Class Template Reference

A tagged union, aka sum type.

```
#include <variant.h>
```

### Public Member Functions

- [Variant](#) ()
- template<typename T , int ix = Traits::template indexOf<T>()>  
[Variant](#) (const T &value)
- [Variant](#) (const [Variant](#) &other)
- [Variant](#) ([Variant](#) &&other) noexcept
- virtual ~[Variant](#) ()
- auto [operator=](#) (const [Variant](#) &other) -> [Variant](#) &
- auto [operator=](#) ([Variant](#) &&other) noexcept -> [Variant](#) &
- template<typename T , int ix = Traits::template indexOf<T>()>  
auto [operator=](#) (const T &value) -> [Variant](#) &  
*sets the variant to one of its member types.*
- template<typename T >  
auto [is](#) () const -> bool  
*checks if T is the current type of this variant.*
- auto [index](#) () const -> int  
*returns the current index of the current state.*
- template<typename T >  
auto [get](#) () -> T \*  
*if the current state is of type T, return it. else null.*
- template<typename T >  
auto [get](#) () const -> const T \*  
*if the current state is of type T, return it. else null.*
- template<int ix>  
auto [get](#) () -> typename Traits::template NthType< ix > \*  
*if the current state is of index ix, return it. else null.*
- template<int ix>  
auto [get](#) () const -> const typename Traits::template NthType< ix > \*  
*if the current state is of index ix, return it. else null.*
- template<typename Visitor >  
auto [visit](#) (const Visitor &f) const -> void  
*visits the variant using a polymorphic functor.*

### 6.61.1 Detailed Description

```
template<typename ... Ts>  
class ticket::Variant< Ts >
```

A tagged union, aka sum type.

This object holds exactly one of its member types, but which type it holds is not statically known. It is entirely on stack, no extra memory allocations are made.

Member types need to be unique and not overlapping.

### 6.61.2 Constructor & Destructor Documentation

**6.61.2.1 Variant()** [1/4] `template<typename ... Ts>`  
`ticket::Variant< Ts >::Variant ( ) [inline]`

**6.61.2.2 Variant()** [2/4] `template<typename ... Ts>`  
`template<typename T , int ix = Traits::template indexOf<T>()>`  
`ticket::Variant< Ts >::Variant (`  
 `const T & value ) [inline]`

constructs the variant from one of its member types.

**6.61.2.3 Variant()** [3/4] `template<typename ... Ts>`  
`ticket::Variant< Ts >::Variant (`  
 `const Variant< Ts > & other ) [inline]`

**6.61.2.4 Variant()** [4/4] `template<typename ... Ts>`  
`ticket::Variant< Ts >::Variant (`  
 `Variant< Ts > && other ) [inline], [noexcept]`

**6.61.2.5 ~Variant()** `template<typename ... Ts>`  
`virtual ticket::Variant< Ts >::~~Variant ( ) [inline], [virtual]`

### 6.61.3 Member Function Documentation

**6.61.3.1 get() [1/4]** `template<typename ... Ts>`  
`template<typename T >`  
`auto ticket::Variant< Ts >::get ( ) -> T * [inline]`

if the current state is of type T, return it. else null.

**6.61.3.2 get() [2/4]** `template<typename ... Ts>`  
`template<int ix>`  
`auto ticket::Variant< Ts >::get ( ) -> typename Traits::template NthType<ix> * [inline]`

if the current state is of index ix, return it. else null.

**6.61.3.3 get() [3/4]** `template<typename ... Ts>`  
`template<typename T >`  
`auto ticket::Variant< Ts >::get ( ) const -> const T * [inline]`

if the current state is of type T, return it. else null.

**6.61.3.4 get() [4/4]** `template<typename ... Ts>`  
`template<int ix>`  
`auto ticket::Variant< Ts >::get ( ) const -> const typename Traits::template NthType<ix> *`  
`[inline]`

if the current state is of index ix, return it. else null.

**6.61.3.5 index()** `template<typename ... Ts>`  
`auto ticket::Variant< Ts >::index ( ) const -> int [inline]`

returns the current index of the current state.

**6.61.3.6 is()** `template<typename ... Ts>`  
`template<typename T >`  
`auto ticket::Variant< Ts >::is ( ) const -> bool [inline]`

checks if T is the current type of this variant.

```
6.61.3.7 operator=() [1/3] template<typename ... Ts>
template<typename T , int ix = Traits::template indexOf<T>()>
auto ticket::Variant< Ts >::operator= (
    const T & value ) -> Variant & [inline]
```

sets the variant to one of its member types.

```
6.61.3.8 operator=() [2/3] template<typename ... Ts>
auto ticket::Variant< Ts >::operator= (
    const Variant< Ts > & other ) -> Variant & [inline]
```

```
6.61.3.9 operator=() [3/3] template<typename ... Ts>
auto ticket::Variant< Ts >::operator= (
    Variant< Ts > && other ) -> Variant & [inline], [noexcept]
```

```
6.61.3.10 visit() template<typename ... Ts>
template<typename Visitor >
auto ticket::Variant< Ts >::visit (
    const Visitor & f ) const -> void [inline]
```

visits the variant using a polymorphic functor.

pass in a polymorphic visitor function, and we will call it with the correct type. If the current type is T, then we would call f(T &). Note that this method deliberately disregards const status. This is to ensure that it still works when this is const.

The documentation for this class was generated from the following file:

- lib/[variant.h](#)

## 6.62 ticket::Vector< T > Class Template Reference

A data container like std::vector.

```
#include <vector.h>
```

### Classes

- class [const\\_iterator](#)
- class [iterator](#)



## Public Member Functions

- `Vector` ()=default
- `Vector` (const `Vector` &other)
- `Vector` (`Vector` &&other) noexcept
- `~Vector` ()
- auto `operator=` (const `Vector` &other) -> `Vector` &
- auto `operator=` (`Vector` &&other) noexcept -> `Vector` &
- auto `at` (const size\_t &pos) -> T &
- auto `at` (const size\_t &pos) const -> const T &
- auto `operator[]` (const size\_t &pos) -> T &
- auto `operator[]` (const size\_t &pos) const -> const T &
- auto `front` () const -> const T &
- auto `back` () const -> const T &
- auto `begin` () -> iterator
- auto `begin` () const -> const\_iterator
- auto `cbegin` () const -> const\_iterator
- auto `end` () -> iterator
- auto `end` () const -> const\_iterator
- auto `cend` () const -> const\_iterator
- auto `empty` () const -> bool
- auto `size` () const -> size\_t
- auto `clear` () -> void
- auto `insert` (iterator pos, const T &value) -> iterator
- auto `insert` (const size\_t &ix, const T &value) -> iterator
- auto `erase` (iterator pos) -> iterator
- auto `erase` (const size\_t &ix) -> iterator
- auto `push_back` (const T &value) -> void
- auto `pop_back` () -> void
- auto `reserve` (size\_t capacity) -> void

### 6.62.1 Detailed Description

```
template<typename T>
class ticket::Vector< T >
```

A data container like std::vector.

store data in a successive memory and support random access.

### 6.62.2 Constructor & Destructor Documentation

**6.62.2.1 Vector()** [1/3] `template<typename T >`  
`ticket::Vector< T >::Vector ( )` [default]

**6.62.2.2 Vector()** [2/3] template<typename T >  
 ticket::Vector< T >::Vector (   
     const Vector< T > & other ) [inline]

**6.62.2.3 Vector()** [3/3] template<typename T >  
 ticket::Vector< T >::Vector (   
     Vector< T > && other ) [inline], [noexcept]

**6.62.2.4 ~Vector()** template<typename T >  
 ticket::Vector< T >::~~Vector ( ) [inline]

### 6.62.3 Member Function Documentation

**6.62.3.1 at()** [1/2] template<typename T >  
 auto ticket::Vector< T >::at (   
     const size\_t & pos ) -> T & [inline]

assigns specified element with bounds checking throw `index_out_of_bound` if pos is not in [0, size)

**6.62.3.2 at()** [2/2] template<typename T >  
 auto ticket::Vector< T >::at (   
     const size\_t & pos ) const -> const T & [inline]

**6.62.3.3 back()** template<typename T >  
 auto ticket::Vector< T >::back ( ) const -> const T & [inline]

access the last element. throw `container_is_empty` if size == 0

**6.62.3.4 begin()** [1/2] template<typename T >  
 auto ticket::Vector< T >::begin ( ) -> iterator [inline]

returns an iterator to the beginning.

**6.62.3.5 begin()** [2/2] template<typename T >  
 auto ticket::Vector< T >::begin ( ) const -> const\_iterator [inline]

**6.62.3.6 cbegin()** `template<typename T >`

```
auto ticket::Vector< T >::cbegin ( ) const -> const_iterator [inline]
```

**6.62.3.7 cend()** `template<typename T >`

```
auto ticket::Vector< T >::cend ( ) const -> const_iterator [inline]
```

**6.62.3.8 clear()** `template<typename T >`

```
auto ticket::Vector< T >::clear ( ) -> void [inline]
```

clears the contents

**6.62.3.9 empty()** `template<typename T >`

```
auto ticket::Vector< T >::empty ( ) const -> bool [inline]
```

checks whether the container is empty

**6.62.3.10 end()** [1/2] `template<typename T >`

```
auto ticket::Vector< T >::end ( ) -> iterator [inline]
```

returns an iterator to the end.

**6.62.3.11 end()** [2/2] `template<typename T >`

```
auto ticket::Vector< T >::end ( ) const -> const_iterator [inline]
```

**6.62.3.12 erase()** [1/2] `template<typename T >`

```
auto ticket::Vector< T >::erase (
    const size_t & ix ) -> iterator [inline]
```

removes the element with index ind. return an iterator pointing to the following element. throw `index_out_of_bound` if `ind >= size`

**6.62.3.13 erase()** [2/2] `template<typename T >`

```
auto ticket::Vector< T >::erase (
    iterator pos ) -> iterator [inline]
```

removes the element at pos. return an iterator pointing to the following element. If the iterator pos refers the last element, the `end()` iterator is returned.

**6.62.3.14 front()** `template<typename T >`

```
auto ticket::Vector< T >::front ( ) const -> const T & [inline]
```

access the first element. throw `container_is_empty` if `size == 0`

**6.62.3.15 insert()** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::insert (`  
`const size_t & ix,`  
`const T & value ) -> iterator [inline]`

inserts value at index ind. after inserting, `this->at(ind) == value` returns an iterator pointing to the inserted value. throw `index_out_of_bound` if `ind > size` (in this situation `ind` can be `size` because after inserting the size will increase 1.)

**6.62.3.16 insert()** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::insert (`  
`iterator pos,`  
`const T & value ) -> iterator [inline]`

inserts value before `pos` returns an iterator pointing to the inserted value.

**6.62.3.17 operator=()** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::operator= (`  
`const Vector< T > & other ) -> Vector & [inline]`

**6.62.3.18 operator=()** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::operator= (`  
`Vector< T > && other ) -> Vector & [inline], [noexcept]`

**6.62.3.19 operator[]()** [1/2] `template<typename T >`  
`auto ticket::Vector< T >::operator[] (`  
`const size_t & pos ) -> T & [inline]`

assigns specified element with bounds checking throw `index_out_of_bound` if `pos` is not in `[0, size)` !!! Pay attentions  
 In STL this operator does not check the boundary but I want you to do.

**6.62.3.20 operator[]()** [2/2] `template<typename T >`  
`auto ticket::Vector< T >::operator[] (`  
`const size_t & pos ) const -> const T & [inline]`

**6.62.3.21 pop\_back()** `template<typename T >`  
`auto ticket::Vector< T >::pop_back ( ) -> void [inline]`

remove the last element from the end. throw `container_is_empty` if `size() == 0`

**6.62.3.22 push\_back()** `template<typename T >`  
`auto ticket::Vector< T >::push_back (`  
`const T & value ) -> void [inline]`

adds an element to the end.

**6.62.3.23 reserve()** `template<typename T >`  
`auto ticket::Vector< T >::reserve (`  
    `size_t capacity ) -> void`   `[inline]`

**6.62.3.24 size()** `template<typename T >`  
`auto ticket::Vector< T >::size ( ) const -> size_t`   `[inline]`

returns the number of elements

The documentation for this class was generated from the following file:

- [lib/vector.h](#)

## 7 File Documentation

### 7.1 lib/algorithm.h File Reference

```
#include <iostream>
#include "utility.h"
```

#### Namespaces

- namespace [ticket](#)

#### Macros

- `#define` [TICKET\\_ALGORIGHM\\_DEFINE\\_BOUND\\_FUNC](#)(name, cf)

#### 7.1.1 Macro Definition Documentation

**7.1.1.1 TICKET\_ALGORIGHM\_DEFINE\_BOUND\_FUNC** `#define TICKET_ALGORIGHM_DEFINE_BOUND_FUNC (`  
    `name,`  
    `cf )`

#### Value:

```
template<class Iterator, class T, class Compare = Less<> \
auto name (Iterator first, Iterator last, const T &value, Compare cmp = {}) -> Iterator { \
    int length = distance(first, last); \
    while (length != 0) { \
        auto it = first; \
        int mid = length / 2; \
        advance(it, mid); \
        if (cmp.cf(value, *it)) { \
            first = ++it; \
            length -= mid + 1; \
        } else { \
            length = mid; \
        } \
    } \
    return first; \
}
```

## 7.2 algorithm.h

[Go to the documentation of this file.](#)

```
1 // This file includes some common algorithms.
2 #ifndef TICKET_LIB_ALGORITHM_H_
3 #define TICKET_LIB_ALGORITHM_H_
4
5 #include <iostream>
6
7 #include "utility.h"
8
9 namespace ticket {
10
11 using std::distance, std::advance;
12
13 #define TICKET_ALGORITHM_DEFINE_BOUND_FUNC(name, cf) \
14 template<class Iterator, class T, class Compare = Less<> \
15 auto name (Iterator first, Iterator last, const T &value, Compare cmp = {}) -> Iterator { \
16     int length = distance(first, last); \
17     while (length != 0) { \
18         auto it = first; \
19         int mid = length / 2; \
20         advance(it, mid); \
21         if (cmp.cf(value, *it)) { \
22             first = ++it; \
23             length -= mid + 1; \
24         } else { \
25             length = mid; \
26         } \
27     } \
28     return first; \
29 }
30 TICKET_ALGORITHM_DEFINE_BOUND_FUNC(upperBound, geq)
31 TICKET_ALGORITHM_DEFINE_BOUND_FUNC(lowerBound, gt)
32 #undef TICKET_ALGORITHM_DEFINE_BOUND_FUNC
33
34 } // namespace ticket
35
36 #endif // TICKET_LIB_ALGORITHM_H_
```

## 7.3 lib/datetime.cpp File Reference

```
#include "datetime.h"
```

### Namespaces

- namespace [ticket](#)

## 7.4 lib/datetime.h File Reference

```
#include <iostream>
```

### Classes

- class [ticket::Date](#)  
*Class representing a date between 2021-06-01 and 2021-08-31 (inclusive).*
- class [ticket::Duration](#)  
*Class representing a length of timespan.*
- class [ticket::Instant](#)  
*Class representing a point of time in a day.*

## Namespaces

- namespace `ticket`

## 7.5 `datetime.h`

[Go to the documentation of this file.](#)

```
1 // This file includes date and time utilities.
2 #ifndef TICKET_LIB_DATETIME_H_
3 #define TICKET_LIB_DATETIME_H_
4
5 #include <iostream>
6
7 namespace ticket {
8
9     class Date {
10     public:
11         Date () = default;
12         Date (int month, int date);
13         explicit Date (const char *str);
14         auto month () const -> int;
15         auto date () const -> int;
16         operator std::string () const;
17         auto operator+ (int dt) const -> Date;
18         auto operator- (int dt) const -> Date;
19         auto operator- (Date rhs) const -> int;
20         auto operator< (const Date &rhs) const -> bool;
21         auto inRange (Date begin, Date end) const -> bool;
22     private:
23         int days_ = 0;
24     };
25
26     class Duration {
27     public:
28         Duration () = default;
29         Duration (int hour, int minute);
30         explicit Duration (int minutes);
31         explicit Duration (const char *str);
32         auto hours () const -> int;
33         auto minutes () const -> int;
34         auto totalMinutes () const -> int;
35         auto operator+ (Duration dt) const -> Duration;
36         auto operator- (Duration dt) const -> Duration;
37         auto operator- () const -> Duration;
38         auto operator< (const Duration &rhs) const -> bool;
39     private:
40         int minutes_ = 0;
41     };
42
43     class Instant {
44     public:
45         Instant () = default;
46         Instant (int hour, int minute);
47         explicit Instant (const char *str);
48         auto daysOverflow () const -> int;
49         auto hour () const -> int;
50         auto minute () const -> int;
51         operator std::string () const;
52         auto operator+ (Duration dt) const -> Instant;
53         auto operator- (Duration dt) const -> Instant;
54         auto operator- (Instant rhs) const -> Duration;
55         auto operator< (const Instant &rhs) const -> bool;
56     private:
57         int minutes_ = 0;
58     };
59 } // namespace ticket
60
61 #endif // TICKET_LIB_DATETIME_H_
```

## 7.6 `lib/exception.h` File Reference

```
#include <iostream>
```

**Classes**

- class `ticket::Exception`  
*The base exception class.*
- class `ticket::IoException`
- class `ticket::OutOfBounds`
- class `ticket::Overflow`
- class `ticket::Underflow`
- class `ticket::NotFound`
- class `ticket::ParseException`

**Namespaces**

- namespace `ticket`

**7.7 exception.h**

[Go to the documentation of this file.](#)

```

1
2
3
4
5
6 #ifndef TICKET_LIB_EXCEPTION_H_
7 #define TICKET_LIB_EXCEPTION_H_
8
9 #include <iostream>
10
11 namespace ticket {
12
13     class Exception : public std::exception {
14     public:
15         Exception () = default;
16         Exception (const char *what) : what_(what) {}
17         virtual ~Exception () = default;
18         virtual auto what () const noexcept -> const char * {
19             return what_;
20         }
21     private:
22         const char * const what_ = "unknown exception";
23     };
24
25     class IoException : public Exception {
26     public:
27         IoException () : Exception("IO exception") {}
28         IoException (const char *what) : Exception(what) {}
29     };
30
31     class OutOfBounds : public Exception {
32     public:
33         OutOfBounds () : Exception("out of bounds") {}
34         OutOfBounds (const char *what) : Exception(what) {}
35     };
36
37     class Overflow : public OutOfBounds {
38     public:
39         Overflow () : OutOfBounds("overflow") {}
40         Overflow (const char *what) : OutOfBounds(what) {}
41     };
42
43     class Underflow : public OutOfBounds {
44     public:
45         Underflow () : OutOfBounds("underflow") {}
46         Underflow (const char *what) : OutOfBounds(what) {}
47     };
48
49     class NotFound : public Exception {
50     public:
51         NotFound () : Exception("underflow") {}
52         NotFound (const char *what) : Exception(what) {}
53     };
54
55     class ParseException : public Exception {
56     public:
57         ParseException () : Exception("parse exception") {}
58         ParseException (const char *what) : Exception(what) {}
59     };
60 } // namespace ticket
61
62 #endif // TICKET_LIB_EXCEPTION_H_

```



## 7.8 lib/file/array.h File Reference

```
#include <cstring>
#include "exception.h"
#include "utility.h"
```

### Classes

- struct `ticket::file::Array< T, maxLength, Cmp >`  
*An on-stack array with utility functions and bound checks.*

### Namespaces

- namespace `ticket`
- namespace `ticket::file`  
*File utilities.*

## 7.9 array.h

[Go to the documentation of this file.](#)

```
1 #ifndef TICKET_LIB_FILE_ARRAY_H_
2 #define TICKET_LIB_FILE_ARRAY_H_
3
4 #include <cstring>
5
6 #include "exception.h"
7 #include "utility.h"
8
9 namespace ticket::file {
10
11 template <typename T, size_t maxLength, typename Cmp = Less<>
12 struct Array {
13 private:
14     auto boundsCheck_ (size_t index) -> void {
15         if (index >= length) throw OutOfBounds("Array: overflow or underflow");
16     }
17     Cmp cmp_;
18 public:
19     size_t length = 0;
20     T content[maxLength];
21     auto indexOf (const T &element) -> size_t {
22         for (size_t i = 0; i < length; ++i) {
23             if (cmp_.equals(element, content[i])) return i;
24         }
25         throw NotFound("Array::indexOf: element not found");
26     }
27     auto includes (const T &element) -> bool {
28         for (size_t i = 0; i < length; ++i) {
29             if (cmp_.equals(element, content[i])) return true;
30         }
31         return false;
32     }
33     auto insert (const T &element, size_t offset) -> void {
34         if (offset != length) boundsCheck_(offset);
35         if (length == maxLength) {
36             throw Overflow("Array::insert: overflow");
37         }
38         if (offset != length) {
39             memmove(
40                 &content[offset + 1],
41                 &content[offset],
42                 (length - offset) * sizeof(content[0])
43             );
44         }
45         content[offset] = element;
46         ++length;
47     }
48 }
49
50 }
```

```

62  auto remove (const T &element) -> void {
63      removeAt(indexOf(element));
64  }
69  auto removeAt (size_t offset) -> void {
70      boundsCheck_(offset);
71      if (offset != length - 1) {
72          memmove(
73              &content[offset],
74              &content[offset + 1],
75              (length - offset - 1) * sizeof(content[0])
76          );
77      }
78      --length;
79  }
81  auto clear () -> void { length = 0; }
82
84  auto copyFrom (
85      const Array &other,
86      size_t ixFrom,
87      size_t ixTo,
88      size_t count
89  ) -> void {
90      if (this == &other) {
91          memmove(
92              &content[ixTo],
93              &content[ixFrom],
94              count * sizeof(content[0])
95          );
96      } else {
97          memcpy(
98              &content[ixTo],
99              &other.content[ixFrom],
100             count * sizeof(content[0])
101          );
102      }
103  }
104
105  auto operator[] (size_t index) -> T & {
106      boundsCheck_(index);
107      return content[index];
108  }
109  auto operator[] (size_t index) const -> const T & {
110      boundsCheck_(index);
111      return content[index];
112  }
113
115  auto pop () -> T {
116      if (length == 0) throw Underflow("Array::pop: underflow");
117      return content[--length];
118  }
120  auto shift () -> T {
121      if (length == 0) throw Underflow("Array::pop: underflow");
122      T result = content[0];
123      removeAt(0);
124      return result;
125  }
127  auto push (const T &object) -> void { insert(object, length); }
129  auto unshift (const T &object) -> void { insert(object, 0); }
130
132  template <typename Functor>
133  auto forEach (const Functor &callback) -> T {
134      for (size_t i = 0; i < length; ++i) callback(content[i]);
135  }
136 };
137
138 } // namespace ticket::file
139
140 #endif // TICKET_LIB_FILE_ARRAY_H_

```

## 7.10 lib/file/bptree.h File Reference

```

#include <cstring>
#include "algorithm.h"
#include "file/array.h"
#include "file/file.h"
#include "file/set.h"
#include "optional.h"
#include "utility.h"
#include "vector.h"

```

## Classes

- class `ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >`  
an implementation of the B+ tree.

## Namespaces

- namespace `ticket`
- namespace `ticket::file`  
*File utilities.*

## 7.11 bptree.h

[Go to the documentation of this file.](#)

```
1 #ifndef TICKET_LIB_FILE_BPTREE_H_
2 #define TICKET_LIB_FILE_BPTREE_H_
3
4 #include <cstring>
5
6 #include "algorithm.h"
7 #include "file/array.h"
8 #include "file/file.h"
9 #include "file/set.h"
10 #include "optional.h"
11 #include "utility.h"
12 #include "vector.h"
13
14 #ifdef TICKET_DEBUG
15 #include <iostream>
16 #endif
17
18 namespace ticket::file {
19
20 template <
21     typename KeyType,
22     typename ValueType,
23     typename CmpKey = Less<>,
24     typename CmpValue = Less<>,
25     typename Meta = Unit,
26     size_t szChunk = kDefaultSzChunk
27 >
28 class BpTree {
29 private:
30     struct Node;
31 public:
32     BpTree (const char *filename) : file_(filename, [this] () { this->init_(); }) {}
33     auto insert (const KeyType &key, const ValueType &value) -> void {
34         Node root = Node::root(*this);
35         insert_({ .key = key, .value = value }, root);
36         if (root.shouldSplit()) split_(root, root, 0);
37         root.update();
38     }
39     auto remove (const KeyType &key, const ValueType &value) -> void {
40         Node root = Node::root(*this);
41         remove_({ .key = key, .value = value }, root);
42         if (root.shouldMerge()) merge_(root, root, 0);
43         root.update();
44     }
45     auto findOne (const KeyType &key) -> Optional<ValueType> {
46         return findOne_(key, Node::root(*this));
47     }
48     auto findMany (const KeyType &key) -> Vector<ValueType> {
49         return findMany_(key, Node::root(*this));
50     }
51     auto findAll () -> Vector<ticket::Pair<KeyType, ValueType>> {
52         return findAll_(Node::root(*this));
53     }
54     auto includes (const KeyType &key, const ValueType &value) -> bool {
55         return includes_({ .key = key, .value = value }, Node::root(*this));
56     }
57     auto getMeta () -> Meta {
58         return file_.getMeta();
59     }
60     auto setMeta (const Meta &meta) -> void {
```

```

91     return file_.setMeta(meta);
92 }
93
100 auto clearCache () -> void { file_.clearCache(); }
101
102 #ifdef TICKET_DEBUG
103 auto print () -> void { print_(Node::root(*this)); }
104 #endif
105
106 private:
107     File<Meta, szChunk> file_;
108     CmpKey cmpKey_;
109     CmpValue cmpValue_;
110
111     // data structures
112
113     struct Pair {
114         KeyType key;
115         ValueType value;
116         auto operator< (const Pair &that) const -> bool {
117             CmpKey cmpKey_;
118             CmpValue cmpValue_;
119             if (!cmpKey_.equals(key, that.key)) return cmpKey_.lt(key, that.key);
120             return cmpValue_.lt(value, that.value);
121         }
122     };
123
124     class KeyComparatorLess_ {
125     public:
126         auto operator() (const Pair &lhs, const KeyType &rhs) -> bool {
127             return cmpKey_.lt(lhs.key, rhs);
128         }
129         auto operator() (const KeyType &lhs, const Pair &rhs) -> bool {
130             return cmpKey_.geq(rhs.key, lhs);
131         }
132     private:
133         CmpKey cmpKey_;
134         CmpValue cmpValue_;
135     };
136
137     using NodeId = unsigned int;
138     // ROOT and INTERMEDIATE nodes are index nodes
139     enum NodeType { kRoot, kIntermediate, kRecord };
140     // if k > kLengthMax, there must be an overflow.
141     static constexpr size_t kLengthMax = 18446744073709000000ULL;
142     struct IndexPayload {
143         static constexpr size_t k = (szChunk - 2 * sizeof(NodeId)) / (sizeof(NodeId) + sizeof(Pair)) / 2 - 1;
144         static_assert(k >= 2 && k < kLengthMax);
145         bool leaf = false;
146         Array<NodeId, 2 * k> children;
147         Set<Pair, 2 * k> splits;
148     };
149
150     struct RecordPayload {
151         static constexpr size_t l = (szChunk - 3 * sizeof(NodeId)) / sizeof(Pair) / 2 - 1;
152         static_assert(l >= 2 && l < kLengthMax);
153         NodeId prev = 0;
154         NodeId next = 0;
155         Set<Pair, 2 * l> entries;
156     };
157     union NodePayload {
158         IndexPayload index;
159         RecordPayload record;
160         NodePayload () {} // NOLINT
161     };
162     struct Node : public ManagedObject<Node, Meta, szChunk> {
163         NodeType type;
164         NodePayload payload;
165         static_assert(sizeof(NodeType) + sizeof(NodePayload) <= szChunk);
166
167         // dynamically type-safe accessors
168         auto leaf () -> bool & { TICKET_ASSERT(type != kRecord); return payload.index.leaf; }
169         auto children () -> Array<NodeId, 2 * IndexPayload::k> & { TICKET_ASSERT(type != kRecord); return payload.index.children; }
170         auto splits () -> Set<Pair, 2 * IndexPayload::k> & { TICKET_ASSERT(type != kRecord); return payload.index.splits; }
171         auto prev () -> NodeId & { TICKET_ASSERT(type == kRecord); return payload.record.prev; }
172         auto next () -> NodeId & { TICKET_ASSERT(type == kRecord); return payload.record.next; }
173         auto entries () -> Set<Pair, 2 * RecordPayload::l> & { TICKET_ASSERT(type == kRecord); return payload.record.entries; }
174
175         Node (BpTree &tree, NodeType type) : ManagedObject<Node, Meta, szChunk>(tree.file_), type(type) {
176             if (type == kRecord) {
177                 new(&payload.record) RecordPayload;
178             } else {
179                 new(&payload.index) IndexPayload;
180             }
181         }
182         ~Node () {

```

```

183     if (type == kRecord) {
184         payload.record.~RecordPayload();
185     } else {
186         payload.index.~IndexPayload();
187     }
188 }
189
190 static auto root (BpTree &tree) -> Node { return Node::get(tree.file_, 0); }
191
192 auto halfLimit () -> size_t {
193     return type == kRecord ? RecordPayload::l : IndexPayload::k;
194 }
195 auto length () -> size_t {
196     return type == kRecord ? payload.record.entries.length : payload.index.children.length;
197 }
198 auto shouldSplit () -> bool { return length() == 2 * halfLimit(); }
199 auto shouldMerge () -> bool { return length() < halfLimit(); }
200 auto lowerBound () -> Pair {
201     return type == kRecord ? payload.record.entries[0] : payload.index.splits[0];
202 }
203 };
204
205 // helper functions
206 auto ixInsert_ (const Pair &entry, Node &node) -> size_t {
207     TICKET_ASSERT(node.type != kRecord);
208     auto &splits = node.splits();
209     size_t ix = upperBound(splits.content, splits.content + splits.length, entry) - splits.content;
210     return ix == 0 ? ix : ix - 1;
211 }
212 auto splitRoot_ (Node &node) -> void {
213     Node left(*this, kIntermediate), right(*this, kIntermediate);
214
215     // copy children and splits
216     left.children().copyFrom(node.children(), 0, 0, IndexPayload::k);
217     left.splits().copyFrom(node.splits(), 0, 0, IndexPayload::k);
218     right.children().copyFrom(node.children(), IndexPayload::k, 0, IndexPayload::k);
219     right.splits().copyFrom(node.splits(), IndexPayload::k, 0, IndexPayload::k);
220     left.children().length = left.splits().length = right.children().length = right.splits().length =
IndexPayload::k;
221
222     // set misc properties and save
223     left.leaf() = right.leaf() = node.leaf();
224     node.leaf() = false;
225     left.save();
226     right.save();
227
228     // initiate the new root node
229     node.children().clear();
230     node.children().insert(left.id(), 0);
231     node.children().insert(right.id(), 1);
232     node.splits().clear();
233     node.splits().insert(left.lowerBound());
234     node.splits().insert(right.lowerBound());
235 }
236 auto split_ (Node &node, Node &parent, size_t ixChild) -> void {
237     TICKET_ASSERT(node.shouldSplit());
238 #ifdef TICKET_DEBUG_BPTREE
239     std::cerr << "[Split] " << node.id() << " (parent " << parent.id() << ")" << std::endl;
240 #endif
241     if (node.type == kRoot) {
242         // the split of the root node is a bit different from other nodes. it produces two extra subnodes.
243         splitRoot_(node);
244         return;
245     }
246     TICKET_ASSERT(node.type != kRoot);
247
248     // create a new next node
249     Node next(*this, node.type);
250     if (node.type == kIntermediate) {
251         next.children().copyFrom(node.children(), IndexPayload::k, 0, IndexPayload::k);
252         next.splits().copyFrom(node.splits(), IndexPayload::k, 0, IndexPayload::k);
253         node.children().length = node.splits().length = next.children().length = next.splits().length =
IndexPayload::k;
254         next.leaf() = node.leaf();
255         next.save();
256     } else {
257         TICKET_ASSERT(node.type == kRecord);
258         next.next() = node.next();
259         next.prev() = node.id();
260         memmove(
261             next.entries().content,
262             &node.entries().content[RecordPayload::l],
263             RecordPayload::l * sizeof(node.entries()[0])
264         );
265         next.entries().length = node.entries().length = RecordPayload::l;
266         next.save();
267         if (next.next() != 0) {

```

```

268     Node nextnext = Node::get(file_, next.next());
269     nextnext.prev() = next.id();
270     nextnext.update();
271 }
272 node.next() = next.id();
273 }
274
275 // update the parent node
276 parent.children().insert(next.id(), ixChild + 1);
277 parent.splits().insert(next.lowerBound());
278 }
279
280 template <typename A, typename B>
281 static auto unshift_(A &to, B &from, size_t k) -> void {
282     // we now have [b[0],...,b[k-1]] and [a[0]...a[k-2]], want a -> [b[0],...,b[k-1],a[0],...,a[k-2]]
283     to.copyFrom(to, 0, k, k - 1);
284     to.copyFrom(from, 0, 0, k);
285     to.length += from.length;
286     from.length = 0;
287 }
288 template <typename A, typename B>
289 static auto push_(A &to, B &from, size_t k) -> void {
290     to.copyFrom(from, 0, k - 1, k);
291     to.length += from.length;
292     from.length = 0;
293 }
294 auto merge_(Node &node, Node &parent, size_t ixChild) -> void {
295     TICKET_ASSERT(node.shouldMerge());
296 #ifdef TICKET_DEBUG_BPTREE
297     std::cerr << "[Merge] " << node.id() << " (parent " << parent.id() << ")" << std::endl;
298 #endif
299     if (node.type == kRoot) {
300         if (node.length() > 1 || node.leaf()) return;
301         Node onlyChild = Node::get(file_, node.children()[0]);
302         memcpy(&node, &onlyChild, sizeof(node));
303         node.type = kRoot;
304         return;
305     }
306     const bool hasPrev = ixChild != 0;
307     const bool hasNext = ixChild != parent.children().length - 1;
308     if (!hasNext) {
309         // don't do anything to the only data node.
310         if (!hasPrev && node.type == kRecord) return;
311         // all index nodes has at least 2 child nodes, except for the root node.
312         TICKET_ASSERT(hasPrev);
313         Node prev = Node::get(file_, parent.children()[ixChild - 1]);
314         if (prev.length() > prev.halfLimit()) {
315             if (node.type == kRecord) {
316                 node.entries().insert(prev.entries().pop());
317             } else {
318                 node.children().unshift(prev.children().pop());
319                 node.splits().insert(prev.splits().pop());
320             }
321             prev.update();
322             parent.splits()[ixChild] = node.lowerBound();
323             return;
324         }
325         TICKET_ASSERT(prev.length() == prev.halfLimit());
326
327         if (node.type == kRecord) {
328             unshift_(node.entries(), prev.entries(), RecordPayload::l);
329             if (prev.prev() != 0) {
330                 Node prevprev = Node::get(file_, prev.prev());
331                 prevprev.next() = node.id();
332                 prevprev.update();
333             }
334             node.prev() = prev.prev();
335         } else {
336             TICKET_ASSERT(node.type == kIntermediate);
337             unshift_(node.children(), prev.children(), IndexPayload::k);
338             unshift_(node.splits(), prev.splits(), IndexPayload::k);
339         }
340         parent.splits()[ixChild] = node.lowerBound();
341         parent.children().removeAt(ixChild - 1);
342         parent.splits().removeAt(ixChild - 1);
343         prev.destroy();
344         return;
345     }
346     TICKET_ASSERT(hasNext);
347
348     // FIXME: remove dupe code here
349     Node next = Node::get(file_, parent.children()[ixChild + 1]);
350     if (next.length() > next.halfLimit()) {
351         if (node.type == kRecord) {
352             node.entries().insert(next.entries().shift());
353         } else {
354             node.children().push(next.children().shift());

```

```

355     node.splits().insert(next.splits().shift());
356 }
357 next.update();
358 parent.splits()[ixChild + 1] = next.lowerBound();
359 return;
360 }
361 TICKET_ASSERT(next.length() == next.halfLimit());
362
363 if (node.type == kRecord) {
364     push_(node.entries(), next.entries(), RecordPayload::l);
365     if (next.next() != 0) {
366         Node nextnext = Node::get(file_, next.next());
367         nextnext.prev() = node.id();
368         nextnext.update();
369     }
370     node.next() = next.next();
371 } else {
372     TICKET_ASSERT(node.type == kIntermediate);
373     push_(node.children(), next.children(), IndexPayload::k);
374     push_(node.splits(), next.splits(), IndexPayload::k);
375 }
376
377 parent.children().removeAt(ixChild + 1);
378 parent.splits().removeAt(ixChild + 1);
379 next.destroy();
380 }
381
382 // FIXME: lengthy function name
383 auto addValuesToVectorForAllKeyFrom_ (Vector<ValueType> &vec, const KeyType &key, Node node, int
first) -> void {
384     // we need to declare i outside to see if we have advanced to the last element
385     int i = first;
386     for (; i < node.length() && cmpKey_.equals(node.entries()[i].key, key); ++i)
vec.push_back(node.entries()[i].value);
387     if (i == node.length() && node.next() != 0) addValuesToVectorForAllKeyFrom_(vec, key,
Node::get(file_, node.next()), 0);
388 }
389 auto addEntriesToVector_ (Vector<ticket::Pair<KeyType, ValueType>> &vec, Node node) -> void {
390     for (int i = 0; i < node.length(); ++i) vec.emplace_back(node.entries()[i].key,
node.entries()[i].value);
391     if (node.next() != 0) addEntriesToVector_(vec, Node::get(file_, node.next()));
392 }
393 auto findFirstChildWithKey_ (const KeyType &key, Node &node) -> ticket::Pair<Node, Optional<Node>> {
394     TICKET_ASSERT(node.type != kRecord);
395     size_t ixGreater = upperBound(
396         node.splits().content,
397         node.splits().content + node.length(),
398         key,
399         Less<KeyComparatorLess_>()
400     ) - node.splits().content;
401     bool hasCdr = ixGreater < node.length() && cmpKey_.equals(node.splits()[ixGreater].key, key);
402     auto cdr = hasCdr ? Optional<Node>(Node::get(file_, node.children()[ixGreater])) :
Optional<Node>(unit);
403     size_t ix = ixGreater == 0 ? ixGreater : ixGreater - 1;
404     return { Node::get(file_, node.children()[ix]), cdr };
405 }
406
407 // operation functions
408 auto insert_ (const Pair &entry, Node &node) -> void {
409     if (node.type == kRecord) {
410         node.entries().insert(entry);
411         TICKET_ASSERT(node.entries().length <= 2 * RecordPayload::l);
412         return;
413     }
414     // if this is the first entry of the root, go create a record node.
415     if (node.children().length == 0) {
416         TICKET_ASSERT(node.type == kRoot);
417         TICKET_ASSERT(node.leaf());
418         Node child(*this, kRecord);
419         child.entries().insert(entry);
420         child.save();
421         node.children().push(child.id());
422         node.splits().insert(entry);
423         return;
424     }
425     size_t ix = ixInsert_(entry, node);
426     if (entry < node.splits()[ix]) node.splits()[ix] = entry;
427     Node nodeToInsert = Node::get(file_, node.children()[ix]);
428     insert_(entry, nodeToInsert);
429     node.splits()[ix] = nodeToInsert.lowerBound();
430     if (nodeToInsert.shouldSplit()) split_(nodeToInsert, node, ix);
431     nodeToInsert.update();
432 }
433 auto remove_ (const Pair &entry, Node &node) -> void {
434     if (node.type == kRecord) {
435         node.entries().remove(entry);
436         return;

```

```

437     }
438     size_t ix = ixInsert_(entry, node);
439     Node child = Node::get(file_, node.children()[ix]);
440     remove_(entry, child);
441     if (child.length() == 0) {
442         TICKET_ASSERT(node.type == kRoot);
443         TICKET_ASSERT(child.type == kRecord);
444         child.destroy();
445         node.children().clear();
446         node.splits().clear();
447         return;
448     }
449     node.splits()[ix] = child.lowerBound();
450     if (child.shouldMerge()) merge_(child, node, ix);
451     child.update();
452 }
453 auto findOne_ (const KeyType &key, Node node) -> Optional<ValueType> {
454     if (node.type != kRecord) {
455         if (node.length() == 0) return unit;
456         auto [ car, cdr ] = findFirstChildWithKey_(key, node);
457         if (!cdr) return findOne_(key, car);
458         auto res = findOne_(key, car);
459         if (res) return res;
460         return findOne_(key, *cdr);
461     }
462     size_t ix = upperBound(
463         node.entries().content,
464         node.entries().content + node.length(),
465         key,
466         Less<KeyComparatorLess_>()
467     ) - node.entries().content;
468     if (ix >= node.length()) return unit;
469     Pair entry = node.entries()[ix];
470     if (!cmpKey_.equals(entry.key, key)) return unit;
471     return entry.value;
472 }
473 auto includes_ (const Pair &entry, Node node) -> bool {
474     if (node.type == kRecord) return node.entries().includes(entry);
475     if (node.length() == 0) return false;
476     return includes_(entry, Node::get(file_, node.children()[ixInsert_(entry, node)]));
477 }
478 auto findMany_ (const KeyType &key, Node node) -> Vector<ValueType> {
479     if (node.type != kRecord) {
480         if (node.length() == 0) return {};
481         auto [ car, cdr ] = findFirstChildWithKey_(key, node);
482         if (!cdr) return findMany_(key, car);
483         Vector<ValueType> res = findMany_(key, car);
484         if (!res.empty()) return res;
485         return findMany_(key, *cdr);
486     }
487     size_t ix = upperBound(
488         node.entries().content,
489         node.entries().content + node.length(),
490         key,
491         Less<KeyComparatorLess_>()
492     ) - node.entries().content;
493     if (ix >= node.length()) return {};
494     Vector<ValueType> res;
495     addValuesToVectorForAllKeyFrom_(res, key, node, ix);
496     return res;
497 }
498 auto findAll_ (Node node) -> Vector<ticket::Pair<KeyType, ValueType> {
499     if (node.type != kRecord) {
500         if (node.length() == 0) return {};
501         return findAll_(Node::get(file_, node.children()[0]));
502     }
503     Vector<ticket::Pair<KeyType, ValueType> res;
504     addEntriesToVector_(res, node);
505     return res;
506 }
507 auto init_ () -> void {
508     Node root(*this, kRoot);
509     root.leaf() = true;
510     root.save();
511     TICKET_ASSERT(root.id() == 0);
512 }
513 #ifdef TICKET_DEBUG
514 auto print_ (Node node) -> void {
515     if (node.type == RECORD) {
516         std::cerr << "[Record " << node.id() << " (" << node.length() << "/" << 2 * RecordPayload::l - 1 << ")]";
517         for (int i = 0; i < node.length(); ++i) std::cerr << " (" << std::string(node.entries()[i].key) << ",
518             " << node.entries()[i].value << ")]";
519         std::cerr << std::endl;
520         return;
521     }
522     std::cerr << "[Node " << node.id() << " (" << node.length() << "/" << 2 * IndexPayload::k - 1 << ")]" <<
523     (node.leaf() ? " leaf" : "") << " ";

```



```

522     for (int i = 0; i < node.length(); ++i) std::cerr << " (" << std::string(node.splits()[i].key) << ", "
    << node.splits()[i].value << ") " << node.children()[i];
523     std::cerr << std::endl;
524     for (int i = 0; i < node.length(); ++i) print_(Node::get(file_, node.children()[i]));
525 }
526 #endif
527 };
528
529 } // namespace ticket::file
530
531 #endif // TICKET_LIB_FILE_BPTREE_H_

```

## 7.12 lib/file/file.h File Reference

```

#include <cstring>
#include <fstream>
#include "hashmap.h"
#include "utility.h"
#include "exception.h"

```

### Classes

- class [ticket::file::File< Meta, szChunk >](#)  
*A chunked file storage with manual garbage collection.*
- class [ticket::file::ManagedObject< T, Meta, szChunk >](#)  
*an opinionated utility base class for the objects to be stored.*

### Namespaces

- namespace [ticket](#)
- namespace [ticket::file](#)  
*File utilities.*

### Variables

- constexpr size\_t [ticket::file::kDefaultSzChunk](#) = 4096

## 7.13 file.h

[Go to the documentation of this file.](#)

```

1 // This file defines several basic file-based utilities.
2 #ifndef TICKET_LIB_FILE_FILE_H_
3 #define TICKET_LIB_FILE_FILE_H_
4
5 #include <cstring>
6 #include <fstream>
7
8 #include "hashmap.h"
9 #include "utility.h"
10 #include "exception.h"
11
12 namespace ticket::file {
13
14     constexpr size_t kDefaultSzChunk = 4096;
15
16     template <typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
17     class File {
18     private:
19         class Metadata;
20     };
21 }

```

```

28 public:
29     template <typename Functor>
30     File (const char *filename, const Functor &initializer) {
31         init_(filename, initializer);
32     }
33     File (const char *filename) {
34         init_(filename, [] {});
35     }
36     ~File () { clearCache(); }
37
38     auto get (void *buf, size_t index, size_t n) -> void {
39         if (index != -1 && cache_.count(index) > 0) {
40             memcpy(buf, cache_[index], n);
41             return;
42         }
43         file_.seekg(offset_(index));
44         file_.read((char *) buf, n);
45         TICKET_ASSERT(file_.good());
46         if (index != -1) putCache_(buf, index, n);
47     }
48     auto set (const void *buf, size_t index, size_t n) -> void {
49         if (index != -1) {
50             // dirty check
51             if (cache_.count(index) > 0 && memcmp(buf, cache_[index], n) == 0) return;
52             putCache_(buf, index, n);
53         }
54         file_.seekp(offset_(index));
55         file_.write((const char *) buf, n);
56         TICKET_ASSERT(file_.good());
57     }
58     auto push (const void *buf, size_t n) -> size_t {
59         Metadata meta = meta_();
60         size_t id = meta.next;
61         if (meta.hasNext) {
62             Metadata nextMeta;
63             get(&nextMeta, meta.next, sizeof(nextMeta));
64             set(&nextMeta, -1, sizeof(nextMeta));
65         } else {
66             ++meta.next;
67             set(&meta, -1, sizeof(meta));
68         }
69         set(buf, id, n);
70         return id;
71     }
72     auto remove (size_t index) -> void {
73         Metadata meta = meta_();
74         set(&meta, index, sizeof(meta));
75         Metadata newMeta(index, true);
76         set(&newMeta, -1, sizeof(newMeta));
77         if (cache_.count(index) > 0) delete[] cache_[index];
78         cache_.erase(cache_.find(index));
79     }
80
81     auto getMeta () -> Meta {
82         return meta_().user;
83     }
84     auto setMeta (const Meta &user) -> void {
85         Metadata meta = meta_();
86         meta.user = user;
87         set(&meta, -1, sizeof(meta));
88     }
89
90     auto clearCache () -> void {
91         for (const auto &[_ , ptr] : cache_) delete[] ptr;
92         cache_.clear();
93     }
94
95 private:
96     struct Metadata {
97         size_t next;
98         bool hasNext;
99         Meta user;
100         Metadata () = default;
101         Metadata (size_t next, bool hasNext) : next(next), hasNext(hasNext) {}
102     };
103     static_assert(sizeof(Metadata) < szChunk);
104
105     template <typename Functor>
106     auto init_ (const char *filename, const Functor &initializer) -> void {
107         bool shouldCreate = false;
108         auto testFile = fopen(filename, "r");
109         if (testFile == nullptr) {
110             if (errno != ENOENT) {
111                 throw IoException("Unable to open file");
112             }
113             shouldCreate = true;
114         } else if (fclose(testFile)) {

```

```

130         throw IOException("Unable to close file");
131     }
132     if (shouldCreate) {
133         auto file = fopen(filename, "w+");
134         if (file == nullptr) {
135             throw IOException("Unable to create file");
136         }
137         if (fclose(file)) {
138             throw IOException("Unable to close file when creating file");
139         }
140     }
141     file_.open(filename);
142     if (!file_.is_open() || !file_.good()) {
143         throw IOException("Unable to open file");
144     }
145     if (shouldCreate) {
146         Metadata meta(0, false);
147         set(&meta, -1, sizeof(meta));
148         initializer();
149     }
150 }
151
152 auto meta_ () -> Metadata {
153     Metadata retval;
154     get(&retval, -1, sizeof(retval));
155     return retval;
156 }
157 auto offset_ (size_t index) -> size_t {
158     return (index + 1) * szChunk;
159 }
160 std::fstream file_;
161 HashMap<size_t, char *> cache_;
162 auto putCache_ (const void *buf, size_t index, size_t n) -> void {
163     char *cache = new char[n];
164     memcpy(cache, buf, n);
165     if (cache_.count(index) > 0) delete[] cache_[index];
166     cache_[index] = cache;
167 }
168 };
169
170 template <typename T, typename Meta = Unit, size_t szChunk = kDefaultSzChunk>
171 class ManagedObject {
172 private:
173     using File_ = File<Meta, szChunk>;
174 public:
175     ManagedObject (File_ &file) : file_(&file) {}
176     virtual ~ManagedObject () = default;
177
178     auto id () -> size_t { return id_; }
179
180     static auto get (File_ &file, size_t id) -> T {
181         char buf[sizeof(T)];
182         file.get(buf, id, sizeof(T));
183         ManagedObject &result = *reinterpret_cast<ManagedObject *>(buf);
184         result.file_ = &file;
185         result.id_ = id;
186         return *reinterpret_cast<T *>(buf);
187     }
188
189     auto save () -> void {
190         if (id_ != -1) throw Exception("Already saved");
191         id_ = file_>push(reinterpret_cast<char *>(this), sizeof(T));
192     }
193     auto update () -> void {
194         if (id_ == -1) throw Exception("Not saved");
195         file_>set(reinterpret_cast<char *>(this), id_, sizeof(T));
196     }
197     auto destroy () -> void {
198         if (id_ == -1) throw Exception("Not saved");
199         file_>remove(id_);
200         id_ = -1;
201     }
202 private:
203     File_ *file_;
204     size_t id_ = -1;
205     ManagedObject (File_ &file, size_t id) : file_(&file), id_(id) {}
206 };
207
208 } // namespace ticket::file
209
210 #endif // TICKET_LIB_FILE_FILE_H_

```

## 7.14 lib/file/index.h File Reference

```
#include "file/bptree.h"
#include "file/varchar.h"
#include "optional.h"
#include "vector.h"
```

### Classes

- class `ticket::file::Index< Key, Model, DataFile >`  
Class representing an index file.
- class `ticket::file::Index< Varchar< maxLength >, Model, DataFile >`  
Specialization of *Index* on *Varchar*.

### Namespaces

- namespace `ticket`
- namespace `ticket::file`  
*File utilities.*

## 7.15 index.h

[Go to the documentation of this file.](#)

```
1 #ifndef TICKET_LIB_FILE_INDEX_H_
2 #define TICKET_LIB_FILE_INDEX_H_
3
4 #include "file/bptree.h"
5 #include "file/varchar.h"
6 #include "optional.h"
7 #include "vector.h"
8
9 namespace ticket::file {
10
11 template <typename Key, typename Model, typename DataFile>
12 class Index {
13 public:
14     Index (Key Model::*ptr, const char *filename, DataFile &datafile)
15         : ptr_(ptr), tree_(filename), file_(datafile) {}
16     auto insert (const Model &model) -> void {
17         tree_.insert(model.*ptr_, model.id());
18     }
19     auto remove (const Model &model) -> void {
20         tree_.remove(model.*ptr_, model.id());
21     }
22     auto findOne (const Key &key) -> Optional<Model> {
23         auto id = tree_.findOne(key);
24         if (!id) return unit;
25         return Model::get(file_, *id);
26     }
27     auto findOneId (const Key &key) -> Optional<int> {
28         return tree_.findOne(key);
29     }
30     auto findMany (const Key &key) -> Vector<Model> {
31         Vector<Model> res;
32         auto ids = tree_.findMany(key);
33         if (ids.size() > 0) res.reserve(ids.size());
34         for (auto id : ids) {
35             res.push_back(Model::get(file_, id));
36         }
37         return res;
38     }
39     auto findManyId (const Key &key) -> Vector<int> {
40         return tree_.findMany(key);
41     }
42 private:
43     Key Model::*ptr_;
```

```

65   BpTree<Key, int> tree_;
66   DataFile &file_;
67 };
68
74 template <size_t maxLength, typename Model, typename DataFile>
75 class Index<Varchar<maxLength>, Model, DataFile> {
76 private:
77     using Key = Varchar<maxLength>;
78 public:
79     Index (Key Model::*ptr, const char *filename, DataFile &datafile)
80         : ptr_(ptr), tree_(filename), file_(datafile) {}
81     auto insert (const Model &model) -> void {
82         tree_.insert(model.ptr_.hash(), model.id());
83     }
84     auto remove (const Model &model) -> void {
85         tree_.remove(model.ptr_.hash(), model.id());
86     }
87     auto findOne (const Key &key) -> Optional<Model> {
88         auto id = tree_.findOne(key.hash());
89         if (!id) return unit;
90         return Model::get(file_, *id);
91     }
92     auto findOneId (const Key &key) -> Optional<int> {
93         return tree_.findOne(key.hash());
94     }
95     auto findMany (const Key &key) -> Vector<Model> {
96         Vector<Model> res;
97         auto ids = tree_.findMany(key.hash());
98         if (ids.size() > 0) res.reserve(ids.size());
99         for (auto id : ids) {
100             res.push_back(Model::get(file_, id));
101         }
102         return res;
103     }
104     auto findManyId (const Key &key) -> Vector<int> {
105         return tree_.findMany(key.hash());
106     }
107 private:
108     Key Model::*ptr_;
109     BpTree<size_t, int> tree_;
110     DataFile &file_;
111 };
112
113 } // namespace ticket::file
114
115 #endif // TICKET_LIB_FILE_INDEX_H_

```

## 7.16 lib/file/set.h File Reference

```

#include <cstring>
#include "algorithm.h"
#include "exception.h"
#include "utility.h"

```

### Classes

- struct [ticket::file::Set< T, maxLength, Cmp >](#)  
A sorted array with utility functions and bound checks.

### Namespaces

- namespace [ticket](#)
- namespace [ticket::file](#)  
*File utilities.*

## 7.17 set.h

[Go to the documentation of this file.](#)

```

1 #ifndef TICKET_LIB_FILE_SET_H_
2 #define TICKET_LIB_FILE_SET_H_
3
4 #include <cstring>
5
6 #include "algorithm.h"
7 #include "exception.h"
8 #include "utility.h"
9
10 // FIXME: remove dupe code of Set and Array. does C++ support mixins?
11 namespace ticket::file {
12
13 template <typename T, size_t maxLength, typename Cmp = Less<>
14 struct Set {
15 private:
16     auto boundsCheck_ (size_t index) -> void {
17         if (index >= length) {
18             throw OutOfBounds("Set: overflow or underflow");
19         }
20     }
21     Cmp cmp_;
22 public:
23     Set () = default;
24     size_t length = 0;
25     T content[maxLength];
26     auto indexOfInsert (const T &element) -> size_t {
27         return lowerBound(content, content + length, element) - content;
28     }
29     auto indexOf (const T &element) -> size_t {
30         size_t index = indexOfInsert(element);
31         if (index >= length || !cmp_.equals(content[index], element)) {
32             throw NotFound("Set::indexOf: element not found");
33         }
34         return index;
35     }
36     auto includes (const T &element) -> bool {
37         size_t ix = indexOfInsert(element);
38         return ix < length && cmp_.equals(content[ix], element);
39     }
40     auto insert (const T &element) -> void {
41         if (length == maxLength) {
42             throw Overflow("Set::insert: overflow");
43         }
44         size_t offset = indexOfInsert(element);
45         if (offset != length) {
46             memmove(
47                 &content[offset + 1],
48                 &content[offset],
49                 (length - offset) * sizeof(content[0])
50             );
51         }
52         content[offset] = element;
53         ++length;
54     }
55     auto remove (const T &element) -> void {
56         removeAt(indexOf(element));
57     }
58     auto removeAt (size_t offset) -> void {
59         boundsCheck_(offset);
60         if (offset != length - 1) {
61             memmove(
62                 &content[offset],
63                 &content[offset + 1],
64                 (length - offset - 1) * sizeof(content[0])
65             );
66         }
67         --length;
68     }
69     auto clear () -> void { length = 0; }
70
71 void copyFrom (const Set &other, size_t ixFrom, size_t ixTo, size_t count) {
72     if (this == &other) {
73         memmove(
74             &content[ixTo],
75             &content[ixFrom],
76             count * sizeof(content[0])
77         );
78     } else {
79         memcpy(
80             &content[ixTo],
81             &other.content[ixFrom],
82             count * sizeof(content[0])
83         );
84     }
85 }
86
87 }
88
89 #endif

```

```

92     );
93 }
94 }
95
96 auto operator[] (size_t index) -> T & {
97     boundsCheck_(index);
98     return content[index];
99 }
100 auto operator[] (size_t index) const -> const T & {
101     boundsCheck_(index);
102     return content[index];
103 }
104
105 auto pop () -> T {
106     if (length == 0) throw Underflow("Set::pop: underflow");
107     return content[--length];
108 }
109
110 auto shift () -> T {
111     if (length == 0) throw Underflow("Set::pop: underflow");
112     T result = content[0];
113     removeAt(0);
114     return result;
115 }
116
117
118 template <typename Functor>
119 auto forEach (const Functor &callback) -> void {
120     for (int i = 0; i < length; ++i) callback(content[i]);
121 }
122 };
123 };
124
125 } // namespace ticket::file
126
127 #endif // TICKET_LIB_FILE_SET_H_

```

## 7.18 lib/file/varchar.h File Reference

```

#include <cstring>
#include <iostream>
#include "exception.h"

```

### Classes

- struct [ticket::file::Varchar< maxLength >](#)  
*A wrapper for const char\* with utility functions and type conversions.*

### Namespaces

- namespace [ticket](#)
- namespace [ticket::file](#)  
*File utilities.*

## 7.19 varchar.h

[Go to the documentation of this file.](#)

```

1 #ifndef TICKET_LIB_FILE_VARCHAR_H_
2 #define TICKET_LIB_FILE_VARCHAR_H_
3
4 #include <cstring>
5 #include <iostream>
6
7 #include "exception.h"
8
9 namespace ticket::file {
10
21 template <int maxLength>

```

```

22 struct Varchar {
23 public:
24     Varchar () { content[0] = '\0'; }
25     Varchar (const std::string &s) {
26         if (s.length() > maxLength) {
27             throw Overflow("Varchar length overflow");
28         }
29         strncpy(content, s.c_str(), maxLength + 1);
30     }
31     Varchar (const char *cstr) : Varchar(std::string(cstr)) {
32         if (strlen(cstr) > maxLength) {
33             throw Overflow("Varchar length overflow");
34         }
35         strncpy(content, cstr, maxLength + 1);
36     }
37
38     template<int A>
39     Varchar (const Varchar<A> &that) { *this = that; }
40     operator std::string () const {
41         return std::string(content);
42     }
43     [[nodiscard]] auto str () const -> std::string {
44         return std::string(*this);
45     }
46
47     auto length () -> int {
48         return strlen(content);
49     }
50
51     template <int A>
52     auto operator= (const Varchar<A> &that) -> Varchar & {
53         if (that.length() > maxLength) {
54             throw Overflow("Varchar length overflow");
55         }
56         strcpy(content, that.content);
57         hash_ = that.hash_;
58         return *this;
59     }
60
61     template <int A>
62     auto operator< (const Varchar<A> &that) const -> bool {
63         return hash() < that.hash();
64     }
65     template <int A>
66     auto operator== (const Varchar<A> &that) const -> bool {
67         return hash() == that.hash();
68     }
69     template <int A>
70     auto operator!= (const Varchar<A> &that) const -> bool {
71         return hash() != that.hash();
72     }
73
74     auto hash () const -> size_t {
75         if (hash_ != 0) return hash_;
76         return hash_ = std::hash<std::string_view>() (content);
77     }
78
79 private:
80     template <int A>
81     friend class Varchar;
82     char content[maxLength + 1];
83     mutable size_t hash_ = 0;
84 };
85
86 } // namespace ticket::file
87
88 #endif // TICKET_LIB_FILE_VARCHAR_H_

```

## 7.20 lib/hashmap.h File Reference

```

#include <functional>
#include <cstdint>
#include "exception.h"
#include "utility.h"
#include "internal/rehash.inc"

```



- ## Namespaces

- ## 7.21 hashmap.h

Generated by Doxygen

```

73     return *this;
74 }
75 auto operator* () const -> reference {
76     return node_>self->value;
77 }
78 auto operator== (const iterator &rhs) const -> bool {
79     return node_ == rhs.node_;
80 }
81 auto operator== (const const_iterator &rhs) const -> bool {
82     return node_ == rhs.node_;
83 }
84 auto operator!= (const iterator &rhs) const -> bool {
85     return !(*this == rhs);
86 }
87 auto operator!= (const const_iterator &rhs) const -> bool {
88     return !(*this == rhs);
89 }
90 auto operator-> () const noexcept -> pointer {
91     return &**this;
92 }
93 private:
94     ListNode *node_;
95     HashMap *home_;
96     friend class const_iterator;
97     friend class HashMap;
98 };
99
100 class const_iterator {
101 public:
102     using difference_type = std::ptrdiff_t;
103     using value_type = const HashMap::value_type;
104     using pointer = value_type *;
105     using reference = value_type &;
106     using iterator_category = std::output_iterator_tag;
107
108     const_iterator () = default;
109     const_iterator (const ListNode *node, const HashMap *home) : node_(node), home_(home) {}
110     const_iterator (const iterator &other) : node_(other.node_), home_(other.home_) {}
111     auto operator++ (int) -> const_iterator {
112         if (node_ == &home_->pivot_) throw Exception("invalid state");
113         auto node = node_;
114         node_ = node->next_;
115         return { node, home_ };
116     }
117     auto operator++ () -> const_iterator & {
118         if (node_ == &home_->pivot_) throw Exception("invalid state");
119         node_ = node->next_;
120         return *this;
121     }
122     auto operator-- (int) -> const_iterator {
123         if (node_ == home_->pivot_.next_) throw Exception("invalid state");
124         auto node = node_;
125         node_ = node->prev_;
126         return { node, home_ };
127     }
128     auto operator-- () -> const_iterator & {
129         if (node_ == home_->pivot_.next_) throw Exception("invalid state");
130         node_ = node->prev_;
131         return *this;
132     }
133     auto operator* () const -> reference {
134         return node_>self->value;
135     }
136     auto operator== (const iterator &rhs) const -> bool {
137         return node_ == rhs.node_;
138     }
139     auto operator== (const const_iterator &rhs) const -> bool {
140         return node_ == rhs.node_;
141     }
142     auto operator!= (const iterator &rhs) const -> bool {
143         return !(*this == rhs);
144     }
145     auto operator!= (const const_iterator &rhs) const -> bool {
146         return !(*this == rhs);
147     }
148     auto operator-> () const noexcept -> pointer {
149         return &**this;
150     }
151 private:
152     const ListNode *node_;
153     const HashMap *home_;
154     friend class iterator;
155     friend class HashMap;
156 };
157
158 HashMap () = default;
159 HashMap (const HashMap &other) { *this = other; }

```

```

160 auto operator= (const HashMap &other) -> HashMap & {
161     if (this == &other) return *this;
162     clear();
163     capacity_ = other.capacity_;
164     size_ = other.size_;
165     store_ = new ListNode[internal::pow2[capacity_]];
166     const ListNode *node = &other.pivot_;
167     for (int i = 0; i < size_; ++i) {
168         node = node->next_;
169         Node *newNode = new Node(*(node->self));
170         int ix = newNode->hash & internal::mask[capacity_];
171         newNode->hashList.insertBefore(&store_[ix]);
172         newNode->iteratorList.insertBefore(&pivot_);
173     }
174     return *this;
175 }
176 ~HashMap () {
177     destroy_();
178 }
179
185 auto at (const Key &key) -> Value & {
186     auto it = find(key);
187     if (it == end()) throw OutOfBounds();
188     return it->second;
189 }
190 auto at (const Key &key) const -> const Value & {
191     return const_cast<HashMap *>(this)->at(key);
192 }
193
199 auto operator[] (const Key &key) -> Value & {
200     return insert({ key, Value() }).first->second;
201 }
202
204 auto operator[] (const Key &key) const -> const Value & { return at(key); }
205
207 auto begin () -> iterator { return { pivot_.next_, this }; }
208 auto cbegin () const -> const_iterator { return { pivot_.next_, this }; }
209
211 auto end () -> iterator { return { &pivot_, this }; }
212 auto cend () const -> const_iterator { return { &pivot_, this }; }
213
215 auto empty () const -> bool {
216     return size_ == 0;
217 }
219 auto size () const -> size_t {
220     return size_;
221 }
222
224 auto clear () -> void {
225     destroy_();
226 }
227
234 auto insert (const value_type &value) -> Pair<iterator, bool> {
235     auto &[ k, _ ] = value;
236     auto hash = hash_(k);
237     if (capacity_ > 0) {
238         int ix = hash & internal::mask[capacity_];
239         if (store_[ix].next() != nullptr) {
240             Node *node = store_[ix].next()->find(k);
241             if (node != nullptr) return { { &node->iteratorList, this }, false };
242         }
243     }
244     growIfNeeded_();
245     int ix = hash & internal::mask[capacity_];
246     Node *node = new Node(value, hash);
247     node->hashList.insertBefore(&store_[ix]);
248     node->iteratorList.insertBefore(&pivot_);
249     ++size_;
250     return { { &node->iteratorList, this }, true };
251 }
252
257 auto erase (iterator pos) -> void {
258     if (pos == end() || pos.home_ != this) throw Exception("invalid state");
259     pos.node->self->hashList.remove();
260     pos.node->self->iteratorList.remove();
261     delete pos.node->self;
262     pos.node_ = &pivot_;
263     --size_;
264 }
265
272 auto count (const Key &key) const -> size_t {
273     return find(key) == cend() ? 0 : 1;
274 }
275
282 auto find (const Key &key) -> iterator {
283     if (empty()) return end();
284     auto ix = hash_(key) & internal::mask[capacity_];

```

```

285     if (store_[ix].next() == nullptr) return end();
286     Node *node = store_[ix].next()->find(key);
287     if (node == nullptr) return end();
288     return { &node->iteratorList, this };
289 }
290 auto find (const Key &key) const -> const_iterator {
291     return const_cast<HashMap *>(this)->find(key);
292 }
293
294 private:
295 struct ListNode {
296     ListNode *prev_ = this;
297     ListNode *next_ = this;
298     auto next () -> Node * { return next_->self; }
299     auto prev () -> Node * { return prev_->self; }
300     Node *self = nullptr;
301     ListNode () = default;
302     ListNode (Node *node) : self(node) {}
303
304     auto insertBefore (ListNode *pivot) -> void {
305         prev_ = pivot->prev_;
306         next_ = pivot;
307         pivot->prev_ = prev_->next_ = this;
308     }
309     auto remove () -> void {
310         prev_->next_ = next_;
311         next_->prev_ = prev_;
312     }
313     auto init () -> void {
314         prev_ = next_ = this;
315     }
316 };
317
318 struct Node {
319     value_type value;
320     unsigned hash;
321     ListNode iteratorList = this, hashList = this;
322     Node () = default;
323     Node (const Node &node) : value(node.value), hash(node.hash) {}
324     Node (const value_type &value, unsigned hash) : value(value), hash(hash) {}
325     auto find (const Key &key) -> Node * {
326         if (Equal()(key, value.first)) return this;
327         if (hashList.next() == nullptr) return nullptr;
328         return hashList.next()->find(key);
329     }
330 };
331 ListNode pivot_;
332 ListNode *store_ = nullptr;
333 int size_ = 0;
334 int capacity_ = 0;
335 constexpr static int kThreshold_ = 2;
336 Hash hash0_;
337 auto hash_ (const Key &key) const -> unsigned {
338     return internal::rehash(hash0_(key));
339 }
340 auto growIfNeeded_ () -> void {
341     auto capacityNeeded = static_cast<unsigned long long>((size_ + 1) * kThreshold_);
342     if (capacityNeeded > internal::pow2[capacity_]) grow();
343 }
344 auto grow_ () -> void {
345     if (capacity_ == 0) {
346         capacity_ = 2;
347         store_ = new ListNode[4];
348         return;
349     }
350     int newCapacity = capacity_ + 1;
351     auto prospective = new ListNode[internal::pow2[newCapacity]];
352     auto node = &pivot_;
353     for (int i = 0; i < size_; ++i) {
354         node = node->next_;
355         int ix = node->self->hash & internal::mask[newCapacity];
356         node->self->hashList.insertBefore(&prospective[ix]);
357     }
358     capacity_ = newCapacity;
359     delete[] store_;
360     store_ = prospective;
361 }
362
363 auto destroy_ () -> void {
364     ListNode *node = pivot_.next_;
365     for (int i = 0; i < size_; ++i) {
366         ListNode *next = node->next_;
367         delete node->self;
368         node = next;
369     }
370     capacity_ = 0;
371     size_ = 0;

```

```

372     delete[] store_;
373     store_ = nullptr;
374     pivot_.init();
375 }
376 };
377
378 } // namespace ticket
379
380 #endif // TICKET_LIB_HASHMAP_H_

```

## 7.22 lib/map.h File Reference

```

#include <cstdint>
#include "internal/tree.h"
#include "utility.h"
#include "exception.h"
#include "internal/map-value-compare.inc"

```

### Classes

- class [ticket::Map< KeyType, ValueType, Compare >](#)  
A sorted key-value map backed by a red-black tree.

### Namespaces

- namespace [ticket](#)

## 7.23 map.h

[Go to the documentation of this file.](#)

```

1 #ifndef TICKET_LIB_MAP_H_
2 #define TICKET_LIB_MAP_H_
3
4 #include <cstdint>
5
6 #include "internal/tree.h"
7 #include "utility.h"
8 #include "exception.h"
9
10 #ifdef DEBUG
11 #include <iostream>
12 #endif
13
14 namespace ticket {
15
16 #include "internal/map-value-compare.inc"
17
18 template <typename KeyType, typename ValueType, typename Compare = internal::LessOp>
19 class Map {
20 public:
21     using value_type = Pair<const KeyType, ValueType>;
22 private:
23     using TreeType = typename internal::RbTree<value_type, internal::MapValueCompare<KeyType, ValueType,
24         Compare>;
25 public:
26     using iterator = typename TreeType::iterator;
27     using const_iterator = typename TreeType::const_iterator;
28
29     Map () = default;
30
31     auto at (const KeyType &key) -> ValueType & {
32         auto it = tree_.find(key);
33         if (it == tree_.end()) throw OutOfBounds();
34         return it->second;
35     }
36
37     auto at (const KeyType &key) const -> const ValueType & {

```

```

41     auto it = tree_.find(key);
42     if (it == tree_.cend()) throw OutOfBounds();
43     return it->second;
44 }
45
46 auto operator[] (const KeyType &key) -> ValueType & {
47     // we need to use the default constructor here. Too bad we have no choice.
48     auto p = tree_.insert({ key, ValueType() });
49     return p.first->second;
50 }
51
52 auto operator[] (const KeyType &key) const -> const ValueType & {
53     return at(key);
54 }
55
56 auto begin () -> iterator {
57     return tree_.begin();
58 }
59
60 auto cbegin () const -> const_iterator {
61     return tree_.cbegin();
62 }
63
64 auto end () -> iterator {
65     return tree_.end();
66 }
67
68 auto cend () const -> const_iterator {
69     return tree_.cend();
70 }
71
72 auto empty () const -> bool {
73     return tree_.empty();
74 }
75
76 auto size () const -> size_t {
77     return tree_.size();
78 }
79
80 auto clear () -> void {
81     tree_.clear();
82 }
83
84 auto insert (const value_type &value) -> Pair<iterator, bool> {
85     return tree_.insert(value);
86 }
87
88 auto erase (iterator pos) -> void {
89     return tree_.erase(pos);
90 }
91
92 auto count (const KeyType &key) const -> size_t {
93     auto it = tree_.find(key);
94     return it == tree_.cend() ? 0 : 1;
95 }
96
97 auto find (const KeyType &key) -> iterator {
98     return tree_.find(key);
99 }
100
101 auto find (const KeyType &key) const -> const_iterator {
102     return tree_.find(key);
103 }
104
105 #ifdef DEBUG
106 auto print () -> void {
107     std::cout << "s=" << size() << " ";
108     for (const auto &p : *this) {
109         std::cout << "(" << p.first.print() << ", " << p.second.print() << ") ";
110     }
111     std::cout << std::endl;
112 }
113 #endif
114
115 private:
116     TreeType tree_;
117 };
118
119 } // namespace ticket
120
121 #endif // TICKET_LIB_MAP_H_

```

## 7.24 lib/optional.h File Reference

```

#include "utility.h"
#include "variant.h"

```

### Classes

- class `ticket::Optional< T >`  
A resemblance of `std::optional`.

## Namespaces

- namespace [ticket](#)

## 7.25 optional.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef TICKET_LIB_OPTIONAL_H_
3 #define TICKET_LIB_OPTIONAL_H_
4
5 #include "utility.h"
6 #include "variant.h"
7
8 namespace ticket {
9
10 template <typename T>
11 class Optional : Variant<Unit, T> {
12 private:
13     using VarT = Variant<Unit, T>;
14 public:
15     Optional () = default;
16     Optional (Unit /* unused */) : VarT(unit) {}
17     Optional (const T &value) : VarT(value) {}
18     auto operator= (const T &value) -> Optional & {
19         VarT::operator=(value);
20         return *this;
21     }
22     operator bool () const {
23         return this->template is<T>();
24     }
25     auto operator* () -> T & {
26         return *this->template get<T>();
27     }
28     auto operator* () const -> const T & {
29         return *this->template get<T>();
30     }
31     auto operator-> () -> T * {
32         return this->template get<T>();
33     }
34     auto operator-> () const -> const T * {
35         return this->template get<T>();
36     }
37 };
38
39 } // namespace ticket
40 #endif // TICKET_LIB_OPTIONAL_H_

```

## 7.26 lib/result.h File Reference

```

#include "utility.h"
#include "variant.h"

```

## Classes

- class [ticket::Result< ResultType, ErrorType >](#)  
*Result<Res, Err> = Res | Err.*

## Namespaces

- namespace [ticket](#)

## 7.27 result.h

[Go to the documentation of this file.](#)

```

1 #ifndef TICKET_LIB_RESULT_H_
2 #define TICKET_LIB_RESULT_H_
3
4 #include "utility.h"
5 #include "variant.h"
6
7 namespace ticket {
8
9     template <typename ResultType, typename ErrorType>
10     class Result : public Variant<ResultType, ErrorType> {
11     public:
12         Result () = delete;
13         template <typename T>
14         Result (const T &value) : Variant<ResultType, ErrorType>(value) {}
15         auto result () -> ResultType & {
16             return *this->template get<ResultType>();
17         }
18         auto result () const -> const ResultType & {
19             return *this->template get<ResultType>();
20         }
21         auto error () -> ErrorType * {
22             return this->template get<ErrorType>();
23         }
24         auto error () const -> const ErrorType * {
25             return this->template get<ErrorType>();
26         }
27         auto success () const -> bool {
28             return this->index() == 0;
29         }
30     };
31 } // namespace ticket
32
33 #endif // TICKET_LIB_RESULT_H_

```

## 7.28 lib/utility.cpp File Reference

```
#include "utility.h"
```

### Namespaces

- namespace [ticket](#)

### Functions

- auto [ticket::split](#) (std::string &str, char sep) -> Vector< std::string\_view >  
*splits the string with sep into several substrings.*
- auto [ticket::copyStrings](#) (const Vector< std::string\_view > &vec) -> Vector< std::string >  
*copies the strings in vec into an array of real strings.*

## 7.29 lib/utility.h File Reference

```

#include <iostream>
#include "vector.h"
#include "internal/cmp.inc"

```



## Classes

- struct `ticket::Unit`  
*An empty class, used at various places.*
- class `ticket::Pair< T1, T2 >`  
*A pair of objects.*
- class `ticket::Cmp< Lt >`  
*Comparison utilities.*

## Namespaces

- namespace `ticket`

## Macros

- `#define TICKET_ASSERT(x)`

## Typedefs

- `template<typename Lt = internal::LessOp>`  
`using ticket::Less = Cmp< Lt >`
- `template<typename Lt = internal::LessOp>`  
`using ticket::Greater = Cmp< internal::GreaterOp< Lt > >`

## Functions

- `auto ticket::split (std::string &str, char sep) -> Vector< std::string_view >`  
*splits the string with sep into several substrings.*
- `auto ticket::copyStrings (const Vector< std::string_view > &vec) -> Vector< std::string >`  
*copies the strings in vec into an array of real strings.*
- `template<typename T >`  
`auto ticket::declval () -> T`  
*declare value, used in type annotations.*
- `template<typename T >`  
`auto ticket::move (T &val) -> T &&`  
*forcefully make an rvalue.*

## Variables

- `constexpr Unit ticket::unit`

### 7.29.1 Macro Definition Documentation

#### 7.29.1.1 TICKET\_ASSERT

```
#define TICKET_ASSERT(  
    x )
```

## 7.30 utility.h

[Go to the documentation of this file.](#)

```

1 // This file defines several common utilities.
2 #ifndef TICKET_LIB_UTILITY_H_
3 #define TICKET_LIB_UTILITY_H_
4
5 #include <iostream>
6
7 #include "vector.h"
8
9 #ifdef TICKET_DEBUG
10 #include <cassert>
11 #define TICKET_ASSERT(x) assert(x)
12 #else
13 #define TICKET_ASSERT(x)
14 #endif // TICKET_DEBUG
15
16 namespace ticket {
17
18     auto split (std::string &str, char sep)
19         -> Vector<std::string_view>;
20
21     auto copyStrings (const Vector<std::string_view> &vec)
22         -> Vector<std::string>;
23
24     struct Unit {
25         constexpr Unit () = default;
26         template <typename T>
27         constexpr Unit (const T & /* unused */) {}
28         auto operator< (const Unit & /* unused */) -> bool {
29             return false;
30         }
31     };
32
33     inline constexpr Unit unit;
34
35     template <typename T>
36     auto declval () -> T;
37
38     template <typename T>
39     auto move (T &val) -> T && {
40         return reinterpret_cast<T &&>(val);
41     }
42
43     template <typename T1, typename T2>
44     class Pair {
45     public:
46         T1 first;
47         T2 second;
48         constexpr Pair () : first(), second() {}
49         Pair (const Pair &other) = default;
50         Pair (Pair &&other) noexcept = default;
51         Pair (const T1 &x, const T2 &y) : first(x), second(y) {}
52         template <class U1, class U2>
53         Pair (U1 &&x, U2 &&y) : first(x), second(y) {}
54         template <class U1, class U2>
55         Pair (const Pair<U1, U2> &other) : first(other.first), second(other.second) {}
56         template <class U1, class U2>
57         Pair (Pair<U1, U2> &&other) : first(other.first), second(other.second) {}
58     };
59
60     template <typename Lt>
61     class Cmp {
62     public:
63         template <typename T, typename U>
64         auto equals (const T &lhs, const U &rhs) -> bool {
65             return !lt_(lhs, rhs) && !lt_(rhs, lhs);
66         }
67         template <typename T, typename U>
68         auto ne (const T &lhs, const U &rhs) -> bool {
69             return !equals(lhs, rhs);
70         }
71         template <typename T, typename U>
72         auto lt (const T &lhs, const U &rhs) -> bool {
73             return lt_(lhs, rhs);
74         }
75         template <typename T, typename U>
76         auto gt (const T &lhs, const U &rhs) -> bool {
77             return lt_(rhs, lhs);
78         }
79         template <typename T, typename U>
80         auto leq (const T &lhs, const U &rhs) -> bool {
81             return !gt(lhs, rhs);
82         }
83         template <typename T, typename U>
84         auto geq (const T &lhs, const U &rhs) -> bool {

```

```

101     return !lt(lhs, rhs);
102 }
103 private:
104     Lt lt_;
105 };
106
107 #include "internal/cmp.inc"
108
109 template <typename Lt = internal::LessOp>
110 using Less = Cmp<Lt>;
111 template <typename Lt = internal::LessOp>
112 using Greater = Cmp<internal::GreaterOp<Lt>>;
113
114 } // namespace ticket
115
116 #endif // TICKET_LIB_UTILITY_H_

```

## 7.31 lib/variant.h File Reference

```

#include "internal/variant-impl.h"
#include "utility.h"

```

### Classes

- class `ticket::Variant< Ts >`  
A tagged union, aka sum type.

### Namespaces

- namespace `ticket`

## 7.32 variant.h

[Go to the documentation of this file.](#)

```

1 #ifndef TICKET_LIB_VARIANT_H_
2 #define TICKET_LIB_VARIANT_H_
3
4 #include "internal/variant-impl.h"
5 #include "utility.h"
6
7 namespace ticket {
8
9     template <typename ...Ts>
10     class Variant {
11     private:
12         using Traits = internal::VariantTraits<Ts...>;
13         using First = typename Traits::template NthType<0>;
14         using Second = typename Traits::template NthType<1>;
15         static constexpr size_t length = sizeof...(Ts);
16         static_assert(length >= 2);
17         static_assert(!Traits::hasDuplicates());
18     public:
19         Variant () : ix_(0), store_(internal::ctorIndex<0>) {}
20         template <typename T, int ix = Traits::template indexOf<T>()>
21         Variant (const T &value) :
22             ix_(ix),
23             store_(internal::ctorIndex<ix>, value) {
24             static_assert(Traits::template includes<T>());
25         }
26         Variant (const Variant &other) {
27             *this = other;
28         }
29         Variant (Variant &&other) noexcept { *this = move(other); }
30         // this class may be extended, so let it be virtual.
31         virtual ~Variant () {
32             destroy_();
33         }
34     };
35 }

```

```

46 auto operator= (const Variant &other) -> Variant & {
47     if (this == &other) return *this;
48     destroy_();
49     ix_ = other.ix_;
50     if constexpr (length == 2) {
51         if (ix_ == 0) new(&get_<First>()) First(other.get_<First>());
52         else new(&get_<Second>()) Second(other.get_<Second>());
53     } else {
54         other.visit([this] (auto &value) {
55             using T = std::remove_cvref_t<decltype(value)>;
56             new(&get_<T>()) T(value);
57         });
58     }
59     return *this;
60 }
61 auto operator= (Variant &&other) noexcept -> Variant & {
62     if (this == &other) return *this;
63     destroy_();
64     ix_ = other.ix_;
65     if constexpr (length == 2) {
66         if (ix_ == 0) new(&get_<First>()) First(move(other.get_<First>()));
67         else new(&get_<Second>()) Second(move(other.get_<Second>()));
68     } else {
69         other.visit([this] (auto &value) {
70             using T = decltype(value);
71             new(&get_<T>()) T(move(value));
72         });
73     }
74     return *this;
75 }
76
77 template <typename T, int ix = Traits::template indexOf<T>()>
78 auto operator= (const T &value) -> Variant & {
79     static_assert(Traits::template includes<T>());
80     destroy_();
81     ix_ = ix;
82     new(&get_<T>()) T(value);
83     return *this;
84 }
85
86
87 template <typename T>
88 auto is () const -> bool {
89     static_assert(Traits::template includes<T>());
90     return ix_ == Traits::template indexOf<T>();
91 }
92
93 auto index () const -> int {
94     return ix_;
95 }
96
97
98 template <typename T>
99 auto get () -> T * {
100     if (is<T>()) return &get_<T>();
101     return nullptr;
102 }
103
104 template <typename T>
105 auto get () const -> const T * {
106     if (is<T>()) return &get_<T>();
107     return nullptr;
108 }
109
110 template <int ix>
111 auto get () -> typename Traits::template NthType<ix> * {
112     if (ix_ != ix) return nullptr;
113     return &get_<typename Traits::template NthType<ix>>();
114 }
115
116 template <int ix>
117 auto get () const -> const typename Traits::template NthType<ix> * {
118     if (ix_ != ix) return nullptr;
119     return &get_<typename Traits::template NthType<ix>>();
120 }
121
122
123 template <typename Visitor>
124 auto visit (const Visitor &f) const -> void {
125     using Vt = typename Traits::template Vtable<Visitor>;
126     // sorry about the C-style cast here... it casts away const.
127     Vt::visit(ix_, f, (void *) &store_);
128 }
129
130 private:
131 int ix_ = -1;
132 typename Traits::Impl store_{internal::ctorValueless};
133
134 template <typename T = void>
135 auto get_ () -> T & {
136     return *reinterpret_cast<T *>(&store_);
137 }
138
139 template <typename T = void>
140 auto get_ () const -> const T & {

```

```

149     return *reinterpret_cast<const T *>(&store_);
150 }
151
152 auto destroy_ () -> void {
153     if (ix_ == -1) return;
154     if constexpr (length == 2) {
155         if (ix_ == 0) get_<First>().~First();
156         else get_<Second>().~Second();
157     } else {
158         visit([] (auto &value) {
159             using T = std::remove_reference_t<decltype(value)>;
160             value.~T();
161         });
162     }
163     ix_ = -1;
164 }
165 };
166
167 } // namespace ticket
168
169 #endif // TICKET_LIB_VARIANT_H_

```

### 7.33 lib/vector.h File Reference

```

#include <climits>
#include <cstddef>
#include <iterator>
#include "exception.h"

```

#### Classes

- class `ticket::Vector< T >`  
*A data container like std::vector.*
- class `ticket::Vector< T >::iterator`
- class `ticket::Vector< T >::const_iterator`

#### Namespaces

- namespace `ticket`

### 7.34 vector.h

[Go to the documentation of this file.](#)

```

1 #ifndef TICKET_LIB_VECTOR_H_
2 #define TICKET_LIB_VECTOR_H_
3
4 #include <climits>
5 #include <cstddef>
6 #include <iterator>
7
8 #include "exception.h"
9
10 namespace ticket {
11
12     template<typename T>
13     class Vector {
14     public:
15         class const_iterator;
16         class iterator {
17         public:
18             using difference_type = std::ptrdiff_t;
19             using value_type = T;
20             using pointer = T *;
21             using reference = T &;
22             using iterator_category = std::output_iterator_tag;

```

```

28
29 private:
30     Vector *home_;
31     pointer ptr_;
32     iterator (Vector *home, pointer ptr) : home_(home), ptr_(ptr) {}
33 public:
34     auto operator+ (const int &n) const -> iterator {
35         return iterator(home_, ptr_ + n);
36     }
37     auto operator- (const int &n) const -> iterator {
38         return iterator(home_, ptr_ - n);
39     }
40     // return the distance between two iterators,
41     // if these two iterators point to different vectors, throw invalid_iterator.
42     auto operator- (const iterator &rhs) const -> int {
43         if (home_ != rhs.home_) throw Exception("invalid operation");
44         return ptr_ - rhs.ptr_;
45     }
46     auto operator+= (const int &n) -> iterator & {
47         ptr_ += n;
48         return *this;
49     }
50     auto operator-= (const int &n) -> iterator & { return (*this += -n); }
51     auto operator++ (int) const -> iterator { return operator+(1); }
52     auto operator++ () -> iterator & { return (*this += 1); }
53     auto operator-- (int) const -> iterator { return operator+(-1); }
54     auto operator-- () -> iterator & { return (*this -= 1); }
55     auto operator* () const -> T & { return *ptr_; }
56     auto operator== (const iterator &rhs) const -> bool { return ptr_ == rhs.ptr_; }
57     auto operator== (const const_iterator &rhs) const -> bool { return ptr_ == rhs.ptr_; }
58     auto operator!= (const iterator &rhs) const -> bool { return !(*this == rhs); }
59     auto operator!= (const const_iterator &rhs) const -> bool { return !(*this == rhs); }
60     auto operator< (const iterator &rhs) const -> bool {
61         return **this < *rhs;
62     }
63     auto operator< (const const_iterator &rhs) const -> bool {
64         return **this < *rhs;
65     }
66     friend class const_iterator;
67     friend class Vector;
68 };
69
70 class const_iterator {
71 public:
72     using difference_type = std::ptrdiff_t;
73     using value_type = T;
74     using pointer = T *;
75     using reference = T &;
76     using iterator_category = std::output_iterator_tag;
77
78 private:
79     const Vector *home_;
80     const T *ptr_;
81     const_iterator (const Vector *home, pointer ptr) : home_(home), ptr_(ptr) {}
82 public:
83     auto operator+ (const int &n) const -> const_iterator {
84         return const_iterator(home_, ptr_ + n);
85     }
86     auto operator- (const int &n) const -> const_iterator {
87         return const_iterator(home_, ptr_ - n);
88     }
89     auto operator- (const const_iterator &rhs) const -> int {
90         if (home_ != rhs.home_) throw Exception("invalid operation");
91         return ptr_ - rhs.ptr_;
92     }
93     auto operator+= (const int &n) -> const_iterator & {
94         ptr_ += n;
95         return *this;
96     }
97     auto operator-= (const int &n) -> const_iterator & { return (*this += -n); }
98     auto operator++ (int) const -> const_iterator { return operator+(1); }
99     auto operator++ () -> const_iterator & { return (*this += 1); }
100     auto operator-- (int) const -> const_iterator { return operator+(-1); }
101     auto operator-- () -> const_iterator & { return (*this -= 1); }
102     auto operator* () const -> const T & { return *ptr_; }
103     auto operator== (const iterator &rhs) const -> bool { return ptr_ == rhs.ptr_; }
104     auto operator== (const const_iterator &rhs) const -> bool { return ptr_ == rhs.ptr_; }
105     auto operator!= (const iterator &rhs) const -> bool { return !(*this == rhs); }
106     auto operator!= (const const_iterator &rhs) const -> bool { return !(*this == rhs); }
107     auto operator< (const iterator &rhs) const -> bool {
108         return **this < *rhs;
109     }
110     auto operator< (const const_iterator &rhs) const -> bool {
111         return **this < *rhs;
112     }
113     friend class iterator;
114     friend class Vector;
115 };

```

```

121 Vector () = default;
122 Vector (const Vector &other) { *this = other; }
123 Vector (Vector &&other) noexcept { *this = move(other); }
124 ~Vector () {
125     destroyContents_();
126     delete[] reinterpret_cast<char *>(storage_);
127 }
128 auto operator= (const Vector &other) -> Vector & {
129     if (this == &other) return *this;
130     clear();
131     grow_(other.capacity_);
132     size_ = other.size_;
133     copyContents_(storage_, other.storage_, size_);
134     return *this;
135 }
136 auto operator= (Vector &&other) noexcept -> Vector & {
137     if (this == &other) return *this;
138     clear();
139     storage_ = other.storage_;
140     size_ = other.size_;
141     capacity_ = other.capacity_;
142     other.size_ = other.capacity_ = 0;
143     other.storage_ = nullptr;
144     return *this;
145 }
146
151 auto at (const size_t &pos) -> T & {
152     checkPosition_(pos);
153     return storage_[pos];
154 }
155 auto at (const size_t &pos) const -> const T & {
156     return const_cast<Vector *>(this)->at(pos);
157 }
158
164 auto operator[] (const size_t &pos) -> T & { return at(pos); }
165 auto operator[] (const size_t &pos) const -> const T & { return at(pos); }
170 auto front () const -> const T & {
171     checkNonEmpty_();
172     return at(0);
173 }
178 auto back () const -> const T & {
179     checkNonEmpty_();
180     return at(size_ - 1);
181 }
185 auto begin () -> iterator {
186     return iterator(this, storage_);
187 }
188 auto begin () const -> const_iterator { return cbegin(); }
189 auto cbegin () const -> const_iterator {
190     return const_iterator(this, storage_);
191 }
195 auto end () -> iterator {
196     return iterator(this, storage_ + size_);
197 }
198 auto end () const -> const_iterator { return cend(); }
199 auto cend () const -> const_iterator {
200     return const_iterator(this, storage_ + size_);
201 }
205 auto empty () const -> bool {
206     return size_ == 0;
207 }
211 auto size () const -> size_t {
212     return size_;
213 }
217 auto clear () -> void {
218     destroyContents_();
219     delete[] reinterpret_cast<char *>(storage_);
220     storage_ = nullptr;
221     capacity_ = 0;
222     size_ = 0;
223 }
228 auto insert (iterator pos, const T &value) -> iterator { return insert(pos.ptr_ - storage_, value); }
235 auto insert (const size_t &ix, const T &value) -> iterator {
236     if (ix > size_) throw OutOfBounds();
237     if (size_ == capacity_) grow_();
238     for (size_t i = size_; i > ix; --i) {
239         storage_[i] = move_(storage_[i - 1]);
240     }
241     storage_[ix] = value;
242     ++size_;
243     return iterator(this, storage_ + ix);
244 }
250 auto erase (iterator pos) -> iterator { return erase(pos.ptr_ - storage_); }
256 auto erase (const size_t &ix) -> iterator {
257     checkPosition_(ix);
258     for (size_t i = ix; i + 1 < size_; ++i) {
259         storage_[i] = move_(storage_[i + 1]);
260     }

```

```

261     (storage_ + size_ - 1)->~T();
262     --size_;
263     return iterator(this, storage_ + ix);
264 }
265
266 auto push_back (const T &value) -> void {
267     if (size_ == capacity_) grow_();
268     new(storage_ + size_) T(value);
269     ++size_;
270 }
271
272 auto pop_back () -> void {
273     checkNonEmpty_();
274     (storage_ + size_ - 1)->~T();
275     --size_;
276 }
277
278 auto reserve (size_t capacity) -> void {
279     if (capacity_ < capacity) grow_(capacity);
280 }
281
282 private:
283     static constexpr size_t kSzDefault_ = 4;
284     static constexpr size_t kSzT_ = sizeof(T);
285     T *storage_ = nullptr;
286     size_t capacity_ = 0;
287     size_t size_ = 0;
288
289     static auto move_ (T &el) -> T && { return reinterpret_cast<T &&>(el); }
290     static auto copyContents_ (T *to, T *from, size_t n) -> void {
291         for (size_t i = 0; i < n; ++i) {
292             to[i] = from[i];
293         }
294     }
295     static auto moveContents_ (T *to, T *from, size_t n) -> void {
296         for (size_t i = 0; i < n; ++i) {
297             new(to + i) T(move_(from[i]));
298             from[i].~T();
299         }
300     }
301     static auto destroyContents_ (T *array, size_t n) -> void {
302         for (size_t i = 0; i < n; ++i) {
303             (array + i)->~T();
304         }
305     }
306     auto destroyContents_ () -> void { destroyContents_(storage_, size_); }
307     auto grow_ (size_t capNew) -> void {
308         T *storeNew = reinterpret_cast<T *>(new char[capNew * kSzT_]);
309         if (storage_ != nullptr) {
310             moveContents_(storeNew, storage_, size_);
311             delete[] reinterpret_cast<char *>(storage_);
312         }
313         storage_ = storeNew;
314         capacity_ = capNew;
315     }
316     auto grow_ () -> void {
317         grow_(storage_ == nullptr ? kSzDefault_ : 2 * capacity_);
318     }
319     auto checkPosition_ (size_t pos) const -> void {
320         // since this is size_t which is unsigned, we could not have pos < 0.
321         if (pos >= size_) throw OutOfBounds();
322     }
323     auto checkNonEmpty_ () const -> void {
324         if (size_ == 0) throw OutOfBounds();
325     }
326 };
327
328 } // namespace ticket
329
330 #endif // TICKET_LIB_VECTOR_H_

```

## 7.35 src/main.cpp File Reference

```

#include <iostream>
#include "parser.h"

```

### Functions

- auto `main()` -> int



### 7.35.1 Function Documentation

**7.35.1.1 main()** `auto main ( ) -> int`

## 7.36 src/misc.cpp File Reference

```
#include "parser.h"
```

### Namespaces

- namespace [ticket](#)

## 7.37 src/order.cpp File Reference

```
#include "order.h"  
#include "parser.h"  
#include "rollback.h"
```

### Namespaces

- namespace [ticket](#)

### Variables

- file::File [ticket::orders](#) {"orders"}
- file::Index< User::Id, Order, decltype(orders)> [ticket::ixOrdersUserId](#) {&Order::user, "orders.user.ix", orders}
- file::File [ticket::pendingOrders](#) {"pending-orders"}
- file::Index< Ride, PendingOrder, decltype(pendingOrders)> [ticket::ixPendingOrdersRide](#)

## 7.38 src/order.h File Reference

```
#include "file/file.h"  
#include "file/index.h"  
#include "train.h"  
#include "user.h"
```

### Classes

- struct [ticket::Order](#)
- struct [ticket::PendingOrder](#)

## Namespaces

- namespace [ticket](#)

## 7.39 order.h

[Go to the documentation of this file.](#)

```
1 #ifndef TICKET_ORDER_H_
2 #define TICKET_ORDER_H_
3
4 #include "file/file.h"
5 #include "file/index.h"
6 #include "train.h"
7 #include "user.h"
8
9 namespace ticket {
10
11 struct Order : public file::ManagedObject<Order> {
12     using Id = int;
13     enum Status { kSuccess, kPending, kRefunded };
14
15     User::Id user;
16     Ride ride;
17     int ixFrom, ixTo;
18     int seats;
19     Status status;
20
22     auto getTrain () -> Train;
23 };
24
25 extern file::File<> orders;
26 extern file::Index<User::Id, Order, decltype(orders)>
27     ixOrdersUserId;
28
29 struct PendingOrder : public file::ManagedObject<PendingOrder> {
30     Ride ride;
31     int ixFrom, ixTo;
32     int seats;
33     Order::Id order;
34
36     auto satisfiable () -> bool;
37     auto getOrder () -> Order;
38 };
39
40 extern file::File<> pendingOrders;
41 extern file::Index<Ride, PendingOrder, decltype(pendingOrders)>
42     ixPendingOrdersRide;
43
44 } // namespace ticket
45
46 #endif // TICKET_ORDER_H_
```

## 7.40 src/parser.cpp File Reference

```
#include "parser.h"
#include "utility.h"
```

## Namespaces

- namespace [ticket](#)
- namespace [ticket::command](#)

*Classes and parsers for commands.*

## Functions

- auto [ticket::command::parse](#) (std::string &str) -> Result< Command, ParseException >  
*parses the command stored in str.*

## 7.41 src/parser.h File Reference

```
#include <iostream>
#include "datetime.h"
#include "exception.h"
#include "optional.h"
#include "variant.h"
#include "result.h"
```

### Classes

- struct [ticket::command::AddUser](#)
- struct [ticket::command::Login](#)
- struct [ticket::command::Logout](#)
- struct [ticket::command::QueryProfile](#)
- struct [ticket::command::ModifyProfile](#)
- struct [ticket::command::AddTrain](#)
- struct [ticket::command::ReleaseTrain](#)
- struct [ticket::command::QueryTrain](#)
- struct [ticket::command::QueryTicket](#)
- struct [ticket::command::QueryTransfer](#)
- struct [ticket::command::BuyTicket](#)
- struct [ticket::command::QueryOrder](#)
- struct [ticket::command::RefundTicket](#)
- struct [ticket::command::Rollback](#)
- struct [ticket::command::Clean](#)
- struct [ticket::command::Exit](#)

### Namespaces

- namespace [ticket](#)
- namespace [ticket::command](#)

*Classes and parsers for commands.*

### Typedefs

- using [ticket::command::Command](#) = Variant< AddUser, Login, Logout, QueryProfile, ModifyProfile, AddTrain, ReleaseTrain, QueryTrain, QueryTicket, QueryTransfer, BuyTicket, QueryOrder, RefundTicket, Rollback, Clean, Exit >

### Enumerations

- enum [ticket::command::SortType](#) { [ticket::command::kTime](#) , [ticket::command::kCost](#) }

## Functions

- auto `ticket::command::parse` (std::string &str) -> Result< Command, ParseException >  
*parses the command stored in str.*
- auto `ticket::command::dispatch` (const AddUser &cmd) -> void  
*Visitor for the commands.*
- auto `ticket::command::dispatch` (const Login &cmd) -> void
- auto `ticket::command::dispatch` (const Logout &cmd) -> void
- auto `ticket::command::dispatch` (const QueryProfile &cmd) -> void
- auto `ticket::command::dispatch` (const ModifyProfile &cmd) -> void
- auto `ticket::command::dispatch` (const AddTrain &cmd) -> void
- auto `ticket::command::dispatch` (const ReleaseTrain &cmd) -> void
- auto `ticket::command::dispatch` (const QueryTrain &cmd) -> void
- auto `ticket::command::dispatch` (const QueryTicket &cmd) -> void
- auto `ticket::command::dispatch` (const QueryTransfer &cmd) -> void
- auto `ticket::command::dispatch` (const BuyTicket &cmd) -> void
- auto `ticket::command::dispatch` (const QueryOrder &cmd) -> void
- auto `ticket::command::dispatch` (const RefundTicket &cmd) -> void
- auto `ticket::command::dispatch` (const Rollback &cmd) -> void
- auto `ticket::command::dispatch` (const Clean &cmd) -> void
- auto `ticket::command::dispatch` (const Exit &cmd) -> void

## 7.42 parser.h

[Go to the documentation of this file.](#)

```
1 // This file is autogenerated. Do not modify.
2 #ifndef TICKET_PARSER_H_
3 #define TICKET_PARSER_H_
4
5 #include <iostream>
6
7 #include "datetime.h"
8 #include "exception.h"
9 #include "optional.h"
10 #include "variant.h"
11 #include "result.h"
12
13 namespace ticket::command {
14
15     enum SortType { kTime, kCost };
16
17     struct AddUser {
18         Optional<std::string> currentUser;
19         std::string username;
20         std::string password;
21         std::string name;
22         std::string email;
23         Optional<int> privilege;
24     };
25
26     struct Login {
27         std::string username;
28         std::string password;
29     };
30
31     struct Logout {
32         std::string username;
33     };
34
35     struct QueryProfile {
36         std::string currentUser;
37         std::string username;
38     };
39
40     struct ModifyProfile {
41         std::string currentUser;
42         std::string username;
43         Optional<std::string> password;
44         Optional<std::string> name;
45         Optional<std::string> email;
46     };
47 }
```

```
47  Optional<int> privilege;
48  };
49
50  struct AddTrain {
51      std::string id;
52      int stops;
53      int seats;
54      Vector<std::string> stations;
55      Vector<int> prices;
56      Instant departure;
57      Vector<Duration> durations;
58      Vector<Duration> stopoverTimes;
59      Vector<Date> dates;
60      char type;
61  };
62
63  struct ReleaseTrain {
64      std::string id;
65  };
66
67  struct QueryTrain {
68      std::string id;
69      Date date;
70  };
71
72  struct QueryTicket {
73      std::string from;
74      std::string to;
75      Date date;
76      SortType sort = kTime;
77  };
78
79  struct QueryTransfer {
80      std::string from;
81      std::string to;
82      Date date;
83      SortType sort = kTime;
84  };
85
86  struct BuyTicket {
87      std::string currentUser;
88      std::string train;
89      Date date;
90      int seats;
91      std::string from;
92      std::string to;
93      bool queue = false;
94  };
95
96  struct QueryOrder {
97      std::string currentUser;
98  };
99
100 struct RefundTicket {
101     std::string currentUser;
102     int index = 1;
103 };
104
105 struct Rollback {
106     int timestamp;
107 };
108
109 struct Clean {
110
111 };
112
113 struct Exit {
114
115 };
116
117
118 using Command = Variant<
119     AddUser,
120     Login,
121     Logout,
122     QueryProfile,
123     ModifyProfile,
124     AddTrain,
125     ReleaseTrain,
126     QueryTrain,
127     QueryTicket,
128     QueryTransfer,
129     BuyTicket,
130     QueryOrder,
131     RefundTicket,
132     Rollback,
133     Clean,
```

```

134     Exit
135 >;
136
142 auto parse (std::string &str)
143     -> Result<Command, ParseException>;
144
155 auto dispatch (const AddUser &cmd) -> void;
156 auto dispatch (const Login &cmd) -> void;
157 auto dispatch (const Logout &cmd) -> void;
158 auto dispatch (const QueryProfile &cmd) -> void;
159 auto dispatch (const ModifyProfile &cmd) -> void;
160 auto dispatch (const AddTrain &cmd) -> void;
161 auto dispatch (const ReleaseTrain &cmd) -> void;
162 auto dispatch (const QueryTrain &cmd) -> void;
163 auto dispatch (const QueryTicket &cmd) -> void;
164 auto dispatch (const QueryTransfer &cmd) -> void;
165 auto dispatch (const BuyTicket &cmd) -> void;
166 auto dispatch (const QueryOrder &cmd) -> void;
167 auto dispatch (const RefundTicket &cmd) -> void;
168 auto dispatch (const Rollback &cmd) -> void;
169 auto dispatch (const Clean &cmd) -> void;
170 auto dispatch (const Exit &cmd) -> void;
171
172 } // namespace ticket::command
173
174 #endif // TICKET_PARSER_H_

```

## 7.43 src/rollback.cpp File Reference

```

#include "rollback.h"
#include "parser.h"

```

### Namespaces

- namespace [ticket](#)

### Variables

- file::File [ticket::logEntries](#) {"rollback-log"}

## 7.44 src/rollback.h File Reference

```

#include "file/file.h"
#include "optional.h"
#include "order.h"
#include "train.h"
#include "user.h"
#include "variant.h"

```

### Classes

- struct [ticket::rollback::AddUser](#)
- struct [ticket::rollback::ModifyProfile](#)
- struct [ticket::rollback::AddTrain](#)
- struct [ticket::rollback::ReleaseTrain](#)
- struct [ticket::rollback::BuyTicket](#)
- struct [ticket::rollback::RefundTicket](#)
- struct [ticket::rollback::LogEntry](#)

## Namespaces

- namespace [ticket](#)
- namespace [ticket::rollback](#)

## Functions

- auto [ticket::rollback::dispatch](#) (const AddUser &log) -> void  
*Visitor for the log entries.*
- auto [ticket::rollback::dispatch](#) (const ModifyProfile &log) -> void
- auto [ticket::rollback::dispatch](#) (const AddTrain &log) -> void
- auto [ticket::rollback::dispatch](#) (const ReleaseTrain &log) -> void
- auto [ticket::rollback::dispatch](#) (const BuyTicket &log) -> void
- auto [ticket::rollback::dispatch](#) (const RefundTicket &log) -> void

## Variables

- file::File [ticket::rollback::logEntries](#)

## 7.45 rollback.h

[Go to the documentation of this file.](#)

```
1 #ifndef TICKET_BACKLOG_H_
2 #define TICKET_BACKLOG_H_
3
4 #include "file/file.h"
5 #include "optional.h"
6 #include "order.h"
7 #include "train.h"
8 #include "user.h"
9 #include "variant.h"
10
11 namespace ticket::rollback {
12
13 struct AddUser {
14     int id;
15 };
16
17 struct ModifyProfile {
18     int id;
19     Optional<User::Password> password;
20     Optional<User::Name> name;
21     Optional<User::Email> email;
22     Optional<User::Privilege> privilege;
23 };
24
25 struct AddTrain {
26     int id;
27 };
28
29 struct ReleaseTrain {
30     int id;
31 };
32
33 struct BuyTicket {
34     int id;
35 };
36
37 struct RefundTicket {
38     int id;
39     Order::Status status;
40 };
41
42 struct LogEntry : public file::ManagedObject<LogEntry> {
43     int timestamp;
44     Variant<
45         AddUser,
46         ModifyProfile,
47         AddTrain,
```

```

48     ReleaseTrain,
49     BuyTicket,
50     RefundTicket
51 > content;
52 };
53
54 extern file::File<> logEntries;
55
56 auto dispatch (const AddUser &log) -> void;
57 auto dispatch (const ModifyProfile &log) -> void;
58 auto dispatch (const AddTrain &log) -> void;
59 auto dispatch (const ReleaseTrain &log) -> void;
60 auto dispatch (const BuyTicket &log) -> void;
61 auto dispatch (const RefundTicket &log) -> void;
62
63 } // namespace ticket::rollback
64
65 #endif // TICKET_BACKLOG_H_

```

## 7.46 src/train.cpp File Reference

```

#include "train.h"
#include "parser.h"
#include "rollback.h"

```

### Namespaces

- namespace [ticket](#)

### Variables

- file::File [ticket::trains](#) {"trains"}
- file::Index< Train::Id, Train, decltype(trains)> [ticket::ixTrainsId](#) {&Train::trainId, "trains.train-id.ix", trains}
- file::BpTree< size\_t, int > [ticket::ixTrainsStop](#) {"trains.stop.ix"}
- file::File [ticket::rideSeats](#) {"ride-seats"}
- file::Index< Ride, RideSeats, decltype(rideSeats)> [ticket::ixRideSeatsRide](#)

## 7.47 src/train.h File Reference

```

#include "datetime.h"
#include "exception.h"
#include "file/array.h"
#include "file/bptree.h"
#include "file/file.h"
#include "file/index.h"
#include "file/varchar.h"
#include "result.h"

```

### Classes

- struct [ticket::Train](#)
- struct [ticket::Train::Stop](#)
- struct [ticket::Train::Edge](#)
- struct [ticket::Ride](#)
- struct [ticket::RideSeats](#)



## Namespaces

- namespace `ticket`
- namespace `ticket::Station`

## Typedefs

- using `ticket::Station::Id` = `file::Varchar< 30 >`

## 7.48 train.h

[Go to the documentation of this file.](#)

```

1 #ifndef TICKET_TRAIN_H_
2 #define TICKET_TRAIN_H_
3
4 #include "datetime.h"
5 #include "exception.h"
6 #include "file/array.h"
7 #include "file/bptree.h"
8 #include "file/file.h"
9 #include "file/index.h"
10 #include "file/varchar.h"
11 #include "result.h"
12
13 namespace ticket {
14
15     namespace Station {
16         using Id = file::Varchar<30>;
17     } // namespace Station
18
19     struct RideSeats;
20
21     struct Train : public file::ManagedObject<Train> {
22         using Id = file::Varchar<20>;
23         using Type = char;
24         struct Stop {
25             Station::Id name;
26         };
27         struct Edge {
28             int price;
29             Instant departure;
30             Instant arrival;
31         };
32
33         Id trainId;
34         file::Array<Stop, 100> stops;
35         file::Array<Edge, 99> edges;
36         int seats;
37         Date begin, end;
38         Type type;
39         bool released;
40
41         auto indexOfStop (const std::string &name) -> Result<int, NotFound>;
42         auto totalPrice (int ixDeparture, int ixArrival) -> int;
43
44         auto getRide (Date date) -> RideSeats;
45         auto getRide (Date date, int ixDeparture) -> RideSeats;
46
47         auto runsOnDate (Date date) -> bool;
48         auto runsOnDate (Date date, int ixDeparture) -> bool;
49     };
50
51     extern file::File<> trains;
52     extern file::Index<Train::Id, Train, decltype(trains)>
53         ixTrainsId;
54     extern file::BpTree<size_t, int> ixTrainsStop;
55
56     struct Ride {
57         int train;
58         Date date;
59
60         auto operator< (const Ride &rhs) const -> bool;
61     };
62
63     struct RideSeats : public file::ManagedObject<RideSeats> {
64         Ride ride;
65         file::Array<int, 99> seatsRemaining;
66     };

```

```

91
97  auto ticketsAvailable (int ixFrom, int ixTo) -> int;
98 };
99
100 extern file::File<> rideSeats;
101 extern file::Index<Ride, RideSeats, decltype(rideSeats)>
102     ixRideSeatsRide;
103
104 } // namespace ticket
105
106 #endif // TICKET_TRAIN_H_

```

## 7.49 src/user.cpp File Reference

```

#include "user.h"
#include <iostream>
#include "hashmap.h"
#include "parser.h"
#include "rollback.h"

```

### Namespaces

- namespace [ticket](#)

### Variables

- file::File [ticket::users](#) {"users"}
- file::Index< User::Id, User, decltype(users)> [ticket::ixUsersUsername](#) {&User::username, "users.↵username.ix", users}
- HashMap< std::string, Unit > [ticket::usersLoggedIn](#)  
*a set of users that are logged in.*

## 7.50 src/user.h File Reference

```

#include "file/file.h"
#include "file/index.h"
#include "file/varchar.h"

```

### Classes

- struct [ticket::User](#)

### Namespaces

- namespace [ticket](#)

## 7.51 user.h

[Go to the documentation of this file.](#)

```
1 #ifndef TICKET_USER_H_
2 #define TICKET_USER_H_
3
4 #include "file/file.h"
5 #include "file/index.h"
6 #include "file/varchar.h"
7
8 namespace ticket {
9
10 struct User : public file::ManagedObject<User> {
11     using Id = file::Varchar<20>;
12     using Password = file::Varchar<30>;
13     using Name = file::Varchar<15>;
14     using Email = file::Varchar<30>;
15     using Privilege = int;
16
17     Id username;
18     Password password;
19     Name name;
20     Email email;
21     Privilege privilege;
22
23     static auto hasUser (const char *username) -> bool;
24 };
25
26
27 extern file::File<> users;
28 extern file::Index<User::Id, User, decltype(users)>
29     ixUsersUsername;
30
31 } // namespace ticket
32
33 #endif // TICKET_USER_H_
```

## Index

~Exception  
  ticket::Exception, [46](#)

~File  
  ticket::file::File< Meta, szChunk >, [48](#)

~HashMap  
  ticket::HashMap< Key, Value, Hash, Equal >, [51](#)

~ManagedObject  
  ticket::file::ManagedObject< T, Meta, szChunk >, [70](#)

~Variant  
  ticket::Variant< Ts >, [111](#)

~Vector  
  ticket::Vector< T >, [115](#)

algorithm.h  
  TICKET\_ALGORIGHM\_DEFINE\_BOUND\_FUNC, [118](#)

arrival  
  ticket::Train::Edge, [45](#)

at  
  ticket::HashMap< Key, Value, Hash, Equal >, [51](#)  
  ticket::Map< KeyType, ValueType, Compare >, [72](#)  
  ticket::Vector< T >, [115](#)

back  
  ticket::Vector< T >, [115](#)

begin  
  ticket::HashMap< Key, Value, Hash, Equal >, [51](#)  
  ticket::Map< KeyType, ValueType, Compare >, [73](#)  
  ticket::Train, [102](#)  
  ticket::Vector< T >, [115](#)

BpTree  
  ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, [27](#)

cbegin  
  ticket::HashMap< Key, Value, Hash, Equal >, [52](#)  
  ticket::Map< KeyType, ValueType, Compare >, [73](#)  
  ticket::Vector< T >, [115](#)

cend  
  ticket::HashMap< Key, Value, Hash, Equal >, [52](#)  
  ticket::Map< KeyType, ValueType, Compare >, [73](#)  
  ticket::Vector< T >, [116](#)

clear  
  ticket::file::Array< T, maxLength, Cmp >, [24](#)  
  ticket::file::Set< T, maxLength, Cmp >, [97](#)  
  ticket::HashMap< Key, Value, Hash, Equal >, [52](#)  
  ticket::Map< KeyType, ValueType, Compare >, [73](#)  
  ticket::Vector< T >, [116](#)

clearCache  
  ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, [27](#)  
  ticket::file::File< Meta, szChunk >, [48](#)

Command  
  ticket::command, [14](#)

const\_iterator  
  ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [34](#)  
  ticket::HashMap< Key, Value, Hash, Equal >::iterator, [63](#)  
  ticket::Map< KeyType, ValueType, Compare >, [72](#)  
  ticket::Vector< T >::iterator, [67](#)

content  
  ticket::file::Array< T, maxLength, Cmp >, [26](#)  
  ticket::file::Set< T, maxLength, Cmp >, [99](#)  
  ticket::rollback::LogEntry, [68](#)

copyFrom  
  ticket::file::Array< T, maxLength, Cmp >, [24](#)  
  ticket::file::Set< T, maxLength, Cmp >, [97](#)

copyStrings  
  ticket, [11](#)

count  
  ticket::HashMap< Key, Value, Hash, Equal >, [52](#)  
  ticket::Map< KeyType, ValueType, Compare >, [73](#)

currentUser  
  ticket::command::AddUser, [22](#)  
  ticket::command::BuyTicket, [29](#)  
  ticket::command::ModifyProfile, [75](#)  
  ticket::command::QueryOrder, [87](#)  
  ticket::command::QueryProfile, [87](#)  
  ticket::command::RefundTicket, [90](#)

Date  
  ticket::Date, [41](#)

date  
  ticket::command::BuyTicket, [30](#)  
  ticket::command::QueryTicket, [88](#)  
  ticket::command::QueryTrain, [89](#)  
  ticket::command::QueryTransfer, [89](#)  
  ticket::Date, [41](#)  
  ticket::Ride, [94](#)

dates  
  ticket::command::AddTrain, [20](#)

daysOverflow  
  ticket::Instant, [59](#)

declval  
  ticket, [11](#)

departure  
  ticket::command::AddTrain, [20](#)  
  ticket::Train::Edge, [45](#)

destroy  
  ticket::file::ManagedObject< T, Meta, szChunk >, [70](#)

difference\_type  
  ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [33](#)  
  ticket::HashMap< Key, Value, Hash, Equal >::iterator, [61](#)  
  ticket::Vector< T >::const\_iterator, [37](#)  
  ticket::Vector< T >::iterator, [65](#)

dispatch

- ticket::command, [15](#), [16](#)
  - ticket::rollback, [18](#), [19](#)
- Duration
  - ticket::Duration, [43](#)
- durations
  - ticket::command::AddTrain, [20](#)
- edges
  - ticket::Train, [102](#)
- Email
  - ticket::User, [106](#)
- email
  - ticket::command::AddUser, [22](#)
  - ticket::command::ModifyProfile, [75](#)
  - ticket::rollback::ModifyProfile, [76](#)
  - ticket::User, [106](#)
- empty
  - ticket::HashMap< Key, Value, Hash, Equal >, [52](#)
  - ticket::Map< KeyType, ValueType, Compare >, [73](#)
  - ticket::Vector< T >, [116](#)
- end
  - ticket::HashMap< Key, Value, Hash, Equal >, [52](#)
  - ticket::Map< KeyType, ValueType, Compare >, [73](#)
  - ticket::Train, [103](#)
  - ticket::Vector< T >, [116](#)
- equals
  - ticket::Cmp< Lt >, [31](#)
- erase
  - ticket::HashMap< Key, Value, Hash, Equal >, [52](#)
  - ticket::Map< KeyType, ValueType, Compare >, [73](#)
  - ticket::Vector< T >, [116](#)
- error
  - ticket::Result< ResultType, ErrorType >, [93](#)
- Exception
  - ticket::Exception, [46](#)
- File
  - ticket::file::File< Meta, szChunk >, [47](#), [48](#)
- find
  - ticket::HashMap< Key, Value, Hash, Equal >, [53](#)
  - ticket::Map< KeyType, ValueType, Compare >, [74](#)
- findAll
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, [28](#)
- findMany
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, [28](#)
  - ticket::file::Index< Key, Model, DataFile >, [55](#)
  - ticket::file::Index< Varchar< maxLength >, Model, DataFile >, [57](#)
- findManyId
  - ticket::file::Index< Key, Model, DataFile >, [55](#)
  - ticket::file::Index< Varchar< maxLength >, Model, DataFile >, [57](#)
- findOne
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, [28](#)
  - ticket::file::Index< Key, Model, DataFile >, [55](#)
  - ticket::file::Index< Varchar< maxLength >, Model, DataFile >, [57](#)
- findOnId
  - ticket::file::Index< Key, Model, DataFile >, [55](#)
  - ticket::file::Index< Varchar< maxLength >, Model, DataFile >, [57](#)
- first
  - ticket::Pair< T1, T2 >, [84](#)
- forEach
  - ticket::file::Array< T, maxLength, Cmp >, [24](#)
  - ticket::file::Set< T, maxLength, Cmp >, [97](#)
- from
  - ticket::command::BuyTicket, [30](#)
  - ticket::command::QueryTicket, [88](#)
  - ticket::command::QueryTransfer, [89](#)
- front
  - ticket::Vector< T >, [116](#)
- geq
  - ticket::Cmp< Lt >, [32](#)
- get
  - ticket::file::File< Meta, szChunk >, [48](#)
  - ticket::file::ManagedObject< T, Meta, szChunk >, [70](#)
  - ticket::Variant< Ts >, [111](#), [112](#)
- getMeta
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, [28](#)
  - ticket::file::File< Meta, szChunk >, [48](#)
- getOrder
  - ticket::PendingOrder, [86](#)
- getRide
  - ticket::Train, [101](#)
- getTrain
  - ticket::Order, [80](#)
- Greater
  - ticket, [10](#)
- gt
  - ticket::Cmp< Lt >, [32](#)
- hash
  - ticket::file::Varchar< maxLength >, [108](#)
- HashMap
  - ticket::HashMap< Key, Value, Hash, Equal >, [51](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [36](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, [64](#)
- hasUser
  - ticket::User, [106](#)
- hour
  - ticket::Instant, [59](#)
- hours
  - ticket::Duration, [44](#)
- Id
  - ticket::Order, [80](#)
  - ticket::Station, [19](#)
  - ticket::Train, [101](#)

- ticket::User, 106
- id
  - ticket::command::AddTrain, 20
  - ticket::command::QueryTrain, 89
  - ticket::command::ReleaseTrain, 91
  - ticket::file::ManagedObject< T, Meta, szChunk >, 70
  - ticket::rollback::AddTrain, 21
  - ticket::rollback::AddUser, 22
  - ticket::rollback::BuyTicket, 30
  - ticket::rollback::ModifyProfile, 76
  - ticket::rollback::RefundTicket, 90
  - ticket::rollback::ReleaseTrain, 91
- includes
  - ticket::file::Array< T, maxLength, Cmp >, 24
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, 28
  - ticket::file::Set< T, maxLength, Cmp >, 97
- Index
  - ticket::file::Index< Key, Model, DataFile >, 54
  - ticket::file::Index< Varchar< maxLength >, Model, DataFile >, 56
- index
  - ticket::command::RefundTicket, 90
  - ticket::Variant< Ts >, 112
- indexOf
  - ticket::file::Array< T, maxLength, Cmp >, 24
  - ticket::file::Set< T, maxLength, Cmp >, 98
- indexOfInsert
  - ticket::file::Set< T, maxLength, Cmp >, 98
- indexOfStop
  - ticket::Train, 102
- inRange
  - ticket::Date, 41
- insert
  - ticket::file::Array< T, maxLength, Cmp >, 25
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, 28
  - ticket::file::Index< Key, Model, DataFile >, 55
  - ticket::file::Index< Varchar< maxLength >, Model, DataFile >, 57
  - ticket::file::Set< T, maxLength, Cmp >, 98
  - ticket::HashMap< Key, Value, Hash, Equal >, 53
  - ticket::Map< KeyType, ValueType, Compare >, 74
  - ticket::Vector< T >, 116, 117
- Instant
  - ticket::Instant, 58, 59
- IOException
  - ticket::IOException, 60
- is
  - ticket::Variant< Ts >, 112
- iterator
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, 36
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, 62
  - ticket::Map< KeyType, ValueType, Compare >, 72
  - ticket::Vector< T >::const\_iterator, 40
- iterator\_category
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, 33
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, 61
  - ticket::Vector< T >::const\_iterator, 37
  - ticket::Vector< T >::iterator, 65
- ixFrom
  - ticket::Order, 80
  - ticket::PendingOrder, 86
- ixOrdersUserId
  - ticket, 11
- ixPendingOrdersRide
  - ticket, 12
- ixRideSeatsRide
  - ticket, 12
- ixTo
  - ticket::Order, 81
  - ticket::PendingOrder, 86
- ixTrainsId
  - ticket, 12
- ixTrainsStop
  - ticket, 12
- ixUsersUsername
  - ticket, 12
- kCost
  - ticket::command, 15
- kDefaultSzChunk
  - ticket::file, 17
- kPending
  - ticket::Order, 80
- kRefunded
  - ticket::Order, 80
- kSuccess
  - ticket::Order, 80
- kTime
  - ticket::command, 15
- length
  - ticket::file::Array< T, maxLength, Cmp >, 26
  - ticket::file::Set< T, maxLength, Cmp >, 99
  - ticket::file::Varchar< maxLength >, 108
- leq
  - ticket::Cmp< Lt >, 32
- Less
  - ticket, 11
- lib/algorithm.h, 118, 119
- lib/datetime.cpp, 119
- lib/datetime.h, 119, 120
- lib/exception.h, 120, 121
- lib/file/array.h, 122
- lib/file/bptree.h, 123, 124
- lib/file/file.h, 130
- lib/file/index.h, 133
- lib/file/set.h, 134, 135
- lib/file/varchar.h, 136
- lib/hashmap.h, 137, 138
- lib/map.h, 142

- lib/optional.h, [143](#), [144](#)
- lib/result.h, [144](#), [145](#)
- lib/utility.cpp, [145](#)
- lib/utility.h, [145](#), [147](#)
- lib/variant.h, [148](#)
- lib/vector.h, [150](#)
- logEntries
  - ticket, [12](#)
  - ticket::rollback, [19](#)
- lt
  - ticket::Cmp< Lt >, [32](#)
- main
  - main.cpp, [154](#)
- main.cpp
  - main, [154](#)
- ManagedObject
  - ticket::file::ManagedObject< T, Meta, szChunk >, [70](#)
- Map
  - ticket::Map< KeyType, ValueType, Compare >, [72](#)
- minute
  - ticket::Instant, [59](#)
- minutes
  - ticket::Duration, [44](#)
- month
  - ticket::Date, [41](#)
- move
  - ticket, [11](#)
- Name
  - ticket::User, [106](#)
- name
  - ticket::command::AddUser, [22](#)
  - ticket::command::ModifyProfile, [75](#)
  - ticket::rollback::ModifyProfile, [76](#)
  - ticket::Train::Stop, [100](#)
  - ticket::User, [106](#)
- ne
  - ticket::Cmp< Lt >, [32](#)
- NotFound
  - ticket::NotFound, [77](#)
- operator bool
  - ticket::Optional< T >, [78](#)
- operator std::string
  - ticket::Date, [41](#)
  - ticket::file::Varchar< maxLength >, [109](#)
  - ticket::Instant, [59](#)
- operator!=
  - ticket::file::Varchar< maxLength >, [109](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [34](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, [62](#)
  - ticket::Vector< T >::const\_iterator, [38](#)
  - ticket::Vector< T >::iterator, [65](#)
- operator<
  - ticket::Date, [42](#)
- ticket::Duration, [44](#)
- ticket::file::Varchar< maxLength >, [109](#)
- ticket::Instant, [59](#)
- ticket::Ride, [94](#)
- ticket::Unit, [105](#)
- ticket::Vector< T >::const\_iterator, [39](#)
- ticket::Vector< T >::iterator, [67](#)
- operator\*
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [35](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, [62](#)
  - ticket::Optional< T >, [78](#), [79](#)
  - ticket::Vector< T >::const\_iterator, [38](#)
  - ticket::Vector< T >::iterator, [65](#)
- operator+
  - ticket::Date, [42](#)
  - ticket::Duration, [44](#)
  - ticket::Instant, [59](#)
  - ticket::Vector< T >::const\_iterator, [38](#)
  - ticket::Vector< T >::iterator, [65](#)
- operator++
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [35](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, [62](#), [63](#)
  - ticket::Vector< T >::const\_iterator, [38](#)
  - ticket::Vector< T >::iterator, [66](#)
- operator+=
  - ticket::Vector< T >::const\_iterator, [38](#)
  - ticket::Vector< T >::iterator, [66](#)
- operator-
  - ticket::Date, [42](#)
  - ticket::Duration, [44](#)
  - ticket::Instant, [59](#)
  - ticket::Vector< T >::const\_iterator, [38](#), [39](#)
  - ticket::Vector< T >::iterator, [66](#)
- operator->
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [35](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, [63](#)
  - ticket::Optional< T >, [79](#)
- operator--
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, [35](#)
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, [63](#)
  - ticket::Vector< T >::const\_iterator, [39](#)
  - ticket::Vector< T >::iterator, [66](#)
- operator-=
  - ticket::Vector< T >::const\_iterator, [39](#)
  - ticket::Vector< T >::iterator, [66](#)
- operator=
  - ticket::file::Varchar< maxLength >, [109](#)
  - ticket::HashMap< Key, Value, Hash, Equal >, [53](#)
  - ticket::Optional< T >, [79](#)
  - ticket::Variant< Ts >, [112](#), [113](#)

- ticket::Vector< T >, 117
- operator==
  - ticket::file::Varchar< maxLength >, 109
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, 35, 36
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, 63
  - ticket::Vector< T >::const\_iterator, 39
  - ticket::Vector< T >::iterator, 67
- operator[]
  - ticket::file::Array< T, maxLength, Cmp >, 25
  - ticket::file::Set< T, maxLength, Cmp >, 98
  - ticket::HashMap< Key, Value, Hash, Equal >, 53
  - ticket::Map< KeyType, ValueType, Compare >, 74
  - ticket::Vector< T >, 117
- Optional
  - ticket::Optional< T >, 78
- order
  - ticket::PendingOrder, 86
- orders
  - ticket, 12
- OutOfBounds
  - ticket::OutOfBounds, 82
- Overflow
  - ticket::Overflow, 82
- Pair
  - ticket::Pair< T1, T2 >, 83, 84
- parse
  - ticket::command, 16
- ParseException
  - ticket::ParseException, 85
- Password
  - ticket::User, 106
- password
  - ticket::command::AddUser, 22
  - ticket::command::Login, 68
  - ticket::command::ModifyProfile, 75
  - ticket::rollback::ModifyProfile, 76
  - ticket::User, 106
- pendingOrders
  - ticket, 12
- pointer
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, 33
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, 61
  - ticket::Vector< T >::const\_iterator, 37
  - ticket::Vector< T >::iterator, 65
- pop
  - ticket::file::Array< T, maxLength, Cmp >, 25
  - ticket::file::Set< T, maxLength, Cmp >, 98
- pop\_back
  - ticket::Vector< T >, 117
- price
  - ticket::Train::Edge, 45
- prices
  - ticket::command::AddTrain, 20
- Privilege
  - ticket::User, 106
- privilege
  - ticket::command::AddUser, 22
  - ticket::command::ModifyProfile, 75
  - ticket::rollback::ModifyProfile, 76
  - ticket::User, 107
- push
  - ticket::file::Array< T, maxLength, Cmp >, 25
  - ticket::file::File< Meta, szChunk >, 48
- push\_back
  - ticket::Vector< T >, 117
- queue
  - ticket::command::BuyTicket, 30
- reference
  - ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, 34
  - ticket::HashMap< Key, Value, Hash, Equal >::iterator, 61
  - ticket::Vector< T >::const\_iterator, 37
  - ticket::Vector< T >::iterator, 65
- released
  - ticket::Train, 103
- remove
  - ticket::file::Array< T, maxLength, Cmp >, 25
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, 29
  - ticket::file::File< Meta, szChunk >, 49
  - ticket::file::Index< Key, Model, DataFile >, 55
  - ticket::file::Index< Varchar< maxLength >, Model, DataFile >, 57
  - ticket::file::Set< T, maxLength, Cmp >, 98
- removeAt
  - ticket::file::Array< T, maxLength, Cmp >, 25
  - ticket::file::Set< T, maxLength, Cmp >, 99
- reserve
  - ticket::Vector< T >, 117
- Result
  - ticket::Result< ResultType, ErrorType >, 92
- result
  - ticket::Result< ResultType, ErrorType >, 93
- ride
  - ticket::Order, 81
  - ticket::PendingOrder, 86
  - ticket::RideSeats, 95
- rideSeats
  - ticket, 13
- runsOnDate
  - ticket::Train, 102
- satisfiable
  - ticket::PendingOrder, 86
- save
  - ticket::file::ManagedObject< T, Meta, szChunk >, 71
- seats
  - ticket::command::AddTrain, 20
  - ticket::command::BuyTicket, 30



- ticket::Order, [81](#)
  - ticket::PendingOrder, [87](#)
  - ticket::Train, [103](#)
- seatsRemaining
  - ticket::RideSeats, [95](#)
- second
  - ticket::Pair< T1, T2 >, [84](#)
- Set
  - ticket::file::Set< T, maxLength, Cmp >, [97](#)
- set
  - ticket::file::File< Meta, szChunk >, [49](#)
- setMeta
  - ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, [29](#)
  - ticket::file::File< Meta, szChunk >, [49](#)
- shift
  - ticket::file::Array< T, maxLength, Cmp >, [26](#)
  - ticket::file::Set< T, maxLength, Cmp >, [99](#)
- size
  - ticket::HashMap< Key, Value, Hash, Equal >, [53](#)
  - ticket::Map< KeyType, ValueType, Compare >, [74](#)
  - ticket::Vector< T >, [118](#)
- sort
  - ticket::command::QueryTicket, [88](#)
  - ticket::command::QueryTransfer, [89](#)
- SortType
  - ticket::command, [14](#)
- split
  - ticket, [11](#)
- src/main.cpp, [153](#)
- src/misc.cpp, [154](#)
- src/order.cpp, [154](#)
- src/order.h, [154](#), [155](#)
- src/parser.cpp, [155](#)
- src/parser.h, [156](#), [157](#)
- src/rollback.cpp, [159](#)
- src/rollback.h, [159](#), [160](#)
- src/train.cpp, [161](#)
- src/train.h, [161](#), [162](#)
- src/user.cpp, [163](#)
- src/user.h, [163](#), [164](#)
- stations
  - ticket::command::AddTrain, [20](#)
- Status
  - ticket::Order, [80](#)
- status
  - ticket::Order, [81](#)
  - ticket::rollback::RefundTicket, [91](#)
- stopoverTimes
  - ticket::command::AddTrain, [20](#)
- stops
  - ticket::command::AddTrain, [21](#)
  - ticket::Train, [103](#)
- str
  - ticket::file::Varchar< maxLength >, [109](#)
- success
  - ticket::Result< ResultType, ErrorType >, [93](#)
- ticket, [9](#)
- copyStrings, [11](#)
- declval, [11](#)
- Greater, [10](#)
- ixOrdersUserId, [11](#)
- ixPendingOrdersRide, [12](#)
- ixRideSeatsRide, [12](#)
- ixTrainsId, [12](#)
- ixTrainsStop, [12](#)
- ixUsersUsername, [12](#)
- Less, [11](#)
- logEntries, [12](#)
- move, [11](#)
- orders, [12](#)
- pendingOrders, [12](#)
- rideSeats, [13](#)
- split, [11](#)
- trains, [13](#)
- unit, [13](#)
- users, [13](#)
- usersLoggedIn, [13](#)
- ticket::Cmp< Lt >, [31](#)
- equals, [31](#)
- geq, [32](#)
- gt, [32](#)
- leq, [32](#)
- lt, [32](#)
- ne, [32](#)
- ticket::command, [13](#)
- Command, [14](#)
- dispatch, [15](#), [16](#)
- kCost, [15](#)
- kTime, [15](#)
- parse, [16](#)
- SortType, [14](#)
- ticket::command::AddTrain, [19](#)
- dates, [20](#)
- departure, [20](#)
- durations, [20](#)
- id, [20](#)
- prices, [20](#)
- seats, [20](#)
- stations, [20](#)
- stopoverTimes, [20](#)
- stops, [21](#)
- type, [21](#)
- ticket::command::AddUser, [21](#)
- currentUser, [22](#)
- email, [22](#)
- name, [22](#)
- password, [22](#)
- privilege, [22](#)
- username, [22](#)
- ticket::command::BuyTicket, [29](#)
- currentUser, [29](#)
- date, [30](#)
- from, [30](#)
- queue, [30](#)
- seats, [30](#)

- to, 30
- train, 30
- ticket::command::Clean, 31
- ticket::command::Exit, 46
- ticket::command::Login, 68
  - password, 68
  - username, 69
- ticket::command::Logout, 69
  - username, 69
- ticket::command::ModifyProfile, 75
  - currentUser, 75
  - email, 75
  - name, 75
  - password, 75
  - privilege, 75
  - username, 75
- ticket::command::QueryOrder, 87
  - currentUser, 87
- ticket::command::QueryProfile, 87
  - currentUser, 87
  - username, 87
- ticket::command::QueryTicket, 88
  - date, 88
  - from, 88
  - sort, 88
  - to, 88
- ticket::command::QueryTrain, 88
  - date, 89
  - id, 89
- ticket::command::QueryTransfer, 89
  - date, 89
  - from, 89
  - sort, 89
  - to, 89
- ticket::command::RefundTicket, 90
  - currentUser, 90
  - index, 90
- ticket::command::ReleaseTrain, 91
  - id, 91
- ticket::command::Rollback, 95
  - timestamp, 96
- ticket::Date, 40
  - Date, 41
  - date, 41
  - inRange, 41
  - month, 41
  - operator std::string, 41
  - operator<, 42
  - operator+, 42
  - operator-, 42
- ticket::Duration, 42
  - Duration, 43
  - hours, 44
  - minutes, 44
  - operator<, 44
  - operator+, 44
  - operator-, 44
  - totalMinutes, 44
- ticket::Exception, 45
  - ~Exception, 46
  - Exception, 46
  - what, 46
- ticket::file, 17
  - kDefaultSzChunk, 17
- ticket::file::Array< T, maxLength, Cmp >, 23
  - clear, 24
  - content, 26
  - copyFrom, 24
  - forEach, 24
  - includes, 24
  - indexOf, 24
  - insert, 25
  - length, 26
  - operator[], 25
  - pop, 25
  - push, 25
  - remove, 25
  - removeAt, 25
  - shift, 26
  - unshift, 26
- ticket::file::BpTree< KeyType, ValueType, CmpKey, CmpValue, Meta, szChunk >, 26
  - BpTree, 27
  - clearCache, 27
  - findAll, 28
  - findMany, 28
  - findOne, 28
  - getMeta, 28
  - includes, 28
  - insert, 28
  - remove, 29
  - setMeta, 29
- ticket::file::File< Meta, szChunk >, 47
  - ~File, 48
  - clearCache, 48
  - File, 47, 48
  - get, 48
  - getMeta, 48
  - push, 48
  - remove, 49
  - set, 49
  - setMeta, 49
- ticket::file::Index< Key, Model, DataFile >, 54
  - findMany, 55
  - findManyId, 55
  - findOne, 55
  - findOneId, 55
  - Index, 54
  - insert, 55
  - remove, 55
- ticket::file::Index< Varchar< maxLength >, Model, DataFile >, 56
  - findMany, 57
  - findManyId, 57
  - findOne, 57
  - findOneId, 57

- Index, 56
- insert, 57
- remove, 57
- ticket::file::ManagedObject< T, Meta, szChunk >, 69
  - ~ManagedObject, 70
  - destroy, 70
  - get, 70
  - id, 70
  - ManagedObject, 70
  - save, 71
  - update, 71
- ticket::file::Set< T, maxLength, Cmp >, 96
  - clear, 97
  - content, 99
  - copyFrom, 97
  - forEach, 97
  - includes, 97
  - indexOf, 98
  - indexOfInsert, 98
  - insert, 98
  - length, 99
  - operator[], 98
  - pop, 98
  - remove, 98
  - removeAt, 99
  - Set, 97
  - shift, 99
- ticket::file::Varchar< maxLength >, 107
  - hash, 108
  - length, 108
  - operator std::string, 109
  - operator!=, 109
  - operator<, 109
  - operator=, 109
  - operator==, 109
  - str, 109
  - Varchar, 108, 109
- ticket::HashMap< Key, Value, Hash, Equal >, 49
  - ~HashMap, 51
  - at, 51
  - begin, 51
  - cbegin, 52
  - cend, 52
  - clear, 52
  - count, 52
  - empty, 52
  - end, 52
  - erase, 52
  - find, 53
  - HashMap, 51
  - insert, 53
  - operator=, 53
  - operator[], 53
  - size, 53
  - value\_type, 50
- ticket::HashMap< Key, Value, Hash, Equal >::const\_iterator, 33
  - const\_iterator, 34
- difference\_type, 33
- HashMap, 36
- iterator, 36
- iterator\_category, 33
- operator!=, 34
- operator\*, 35
- operator++, 35
- operator->, 35
- operator--, 35
- operator==, 35, 36
- pointer, 33
- reference, 34
- value\_type, 34
- ticket::HashMap< Key, Value, Hash, Equal >::iterator, 60
  - const\_iterator, 63
  - difference\_type, 61
  - HashMap, 64
  - iterator, 62
  - iterator\_category, 61
  - operator!=, 62
  - operator\*, 62
  - operator++, 62, 63
  - operator->, 63
  - operator--, 63
  - operator==, 63
  - pointer, 61
  - reference, 61
  - value\_type, 62
- ticket::Instant, 58
  - daysOverflow, 59
  - hour, 59
  - Instant, 58, 59
  - minute, 59
  - operator std::string, 59
  - operator<, 59
  - operator+, 59
  - operator-, 59
- ticket::IOException, 60
  - IOException, 60
- ticket::Map< KeyType, ValueType, Compare >, 71
  - at, 72
  - begin, 73
  - cbegin, 73
  - cend, 73
  - clear, 73
  - const\_iterator, 72
  - count, 73
  - empty, 73
  - end, 73
  - erase, 73
  - find, 74
  - insert, 74
  - iterator, 72
  - Map, 72
  - operator[], 74
  - size, 74
  - value\_type, 72

- ticket::NotFound, 76
  - NotFound, 77
- ticket::Optional< T >, 77
  - operator bool, 78
  - operator\*, 78, 79
  - operator->, 79
  - operator=, 79
  - Optional, 78
- ticket::Order, 79
  - getTrain, 80
  - Id, 80
  - ixFrom, 80
  - ixTo, 81
  - kPending, 80
  - kRefunded, 80
  - kSuccess, 80
  - ride, 81
  - seats, 81
  - Status, 80
  - status, 81
  - user, 81
- ticket::OutOfBounds, 81
  - OutOfBounds, 82
- ticket::Overflow, 82
  - Overflow, 82
- ticket::Pair< T1, T2 >, 83
  - first, 84
  - Pair, 83, 84
  - second, 84
- ticket::ParseException, 85
  - ParseException, 85
- ticket::PendingOrder, 85
  - getOrder, 86
  - ixFrom, 86
  - ixTo, 86
  - order, 86
  - ride, 86
  - satisfiable, 86
  - seats, 87
- ticket::Result< ResultType, ErrorType >, 92
  - error, 93
  - Result, 92
  - result, 93
  - success, 93
- ticket::Ride, 93
  - date, 94
  - operator<, 94
  - train, 94
- ticket::RideSeats, 94
  - ride, 95
  - seatsRemaining, 95
  - ticketsAvailable, 95
- ticket::rollback, 18
  - dispatch, 18, 19
  - logEntries, 19
- ticket::rollback::AddTrain, 21
  - id, 21
- ticket::rollback::AddUser, 22
  - id, 22
- ticket::rollback::BuyTicket, 30
  - id, 30
- ticket::rollback::LogEntry, 68
  - content, 68
  - timestamp, 68
- ticket::rollback::ModifyProfile, 76
  - email, 76
  - id, 76
  - name, 76
  - password, 76
  - privilege, 76
- ticket::rollback::RefundTicket, 90
  - id, 90
  - status, 91
- ticket::rollback::ReleaseTrain, 91
  - id, 91
- ticket::Station, 19
  - Id, 19
- ticket::Train, 100
  - begin, 102
  - edges, 102
  - end, 103
  - getRide, 101
  - Id, 101
  - indexOfStop, 102
  - released, 103
  - runsOnDate, 102
  - seats, 103
  - stops, 103
  - totalPrice, 102
  - trainId, 103
  - Type, 101
  - type, 103
- ticket::Train::Edge, 45
  - arrival, 45
  - departure, 45
  - price, 45
- ticket::Train::Stop, 99
  - name, 100
- ticket::Underflow, 103
  - Underflow, 104
- ticket::Unit, 104
  - operator<, 105
  - Unit, 104
- ticket::User, 105
  - Email, 106
  - email, 106
  - hasUser, 106
  - Id, 106
  - Name, 106
  - name, 106
  - Password, 106
  - password, 106
  - Privilege, 106
  - privilege, 107
  - username, 107
- ticket::Variant< Ts >, 110

- ~Variant, 111
- get, 111, 112
- index, 112
- is, 112
- operator=, 112, 113
- Variant, 111
- visit, 113
- ticket::Vector< T >, 113
  - ~Vector, 115
  - at, 115
  - back, 115
  - begin, 115
  - cbegin, 115
  - cend, 116
  - clear, 116
  - empty, 116
  - end, 116
  - erase, 116
  - front, 116
  - insert, 116, 117
  - operator=, 117
  - operator[], 117
  - pop\_back, 117
  - push\_back, 117
  - reserve, 117
  - size, 118
  - Vector, 114, 115
- ticket::Vector< T >::const\_iterator, 36
  - difference\_type, 37
  - iterator, 40
  - iterator\_category, 37
  - operator!=, 38
  - operator<, 39
  - operator\*, 38
  - operator+, 38
  - operator++, 38
  - operator+=, 38
  - operator-, 38, 39
  - operator--, 39
  - operator-=, 39
  - operator==, 39
  - pointer, 37
  - reference, 37
  - value\_type, 37
  - Vector, 40
- ticket::Vector< T >::iterator, 64
  - const\_iterator, 67
  - difference\_type, 65
  - iterator\_category, 65
  - operator!=, 65
  - operator<, 67
  - operator\*, 65
  - operator+, 65
  - operator++, 66
  - operator+=, 66
  - operator-, 66
  - operator--, 66
  - operator-=, 66
- operator==, 67
- pointer, 65
- reference, 65
- value\_type, 65
- Vector, 67
- TICKET\_ALGORIGHM\_DEFINE\_BOUND\_FUNC
  - algorithm.h, 118
- TICKET\_ASSERT
  - utility.h, 146
- ticketsAvailable
  - ticket::RideSeats, 95
- timestamp
  - ticket::command::Rollback, 96
  - ticket::rollback::LogEntry, 68
- to
  - ticket::command::BuyTicket, 30
  - ticket::command::QueryTicket, 88
  - ticket::command::QueryTransfer, 89
- totalMinutes
  - ticket::Duration, 44
- totalPrice
  - ticket::Train, 102
- train
  - ticket::command::BuyTicket, 30
  - ticket::Ride, 94
- trainId
  - ticket::Train, 103
- trains
  - ticket, 13
- Type
  - ticket::Train, 101
- type
  - ticket::command::AddTrain, 21
  - ticket::Train, 103
- Underflow
  - ticket::Underflow, 104
- Unit
  - ticket::Unit, 104
- unit
  - ticket, 13
- unshift
  - ticket::file::Array< T, maxLength, Cmp >, 26
- update
  - ticket::file::ManagedObject< T, Meta, szChunk >, 71
- user
  - ticket::Order, 81
- username
  - ticket::command::AddUser, 22
  - ticket::command::Login, 69
  - ticket::command::Logout, 69
  - ticket::command::ModifyProfile, 75
  - ticket::command::QueryProfile, 87
  - ticket::User, 107
- users
  - ticket, 13
- usersLoggedIn
  - ticket, 13

## utility.h

TICKET\_ASSERT, [146](#)

## value\_type

ticket::HashMap< Key, Value, Hash, Equal >, [50](#)ticket::HashMap< Key, Value, Hash, Equal  
>::const\_iterator, [34](#)ticket::HashMap< Key, Value, Hash, Equal  
>::iterator, [62](#)ticket::Map< KeyType, ValueType, Compare >, [72](#)ticket::Vector< T >::const\_iterator, [37](#)ticket::Vector< T >::iterator, [65](#)

## Varchar

ticket::file::Varchar< maxLength >, [108](#), [109](#)

## Variant

ticket::Variant< Ts >, [111](#)

## Vector

ticket::Vector< T >, [114](#), [115](#)ticket::Vector< T >::const\_iterator, [40](#)ticket::Vector< T >::iterator, [67](#)

## visit

ticket::Variant< Ts >, [113](#)

## what

ticket::Exception, [46](#)