Primus

Sascha Meissner und Mats Bechtold
Version 1.0

Table of Contents

Table of Contents

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

File Index

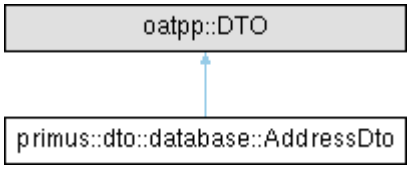
File List

Here is a list of all documented files with brief descriptions:

Class Documentation

primus::dto::database::AddressDto Class Reference

Inheritance diagram for primus::dto::database::AddressDto:



Detailed Description

Definition at line **20** of file **DatabaseDtos.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/DatabaseDtos.hpp

primus::component::AppComponent Class Reference

`#include <AppComponent.hpp>`

Public Member Functions

- **OATPP_CREATE_COMPONENT** (std::shared_ptr< oatpp::network::ServerConnectionProvider >, serverConnectionProvider)([]
- **OATPP_CREATE_COMPONENT** (std::shared_ptr< oatpp::web::server::HttpRouter >, httpRouter)([]
- **OATPP_CREATE_COMPONENT** (std::shared_ptr< oatpp::network::ConnectionHandler >, serverConnectionHandler)([]
- **OATPP_CREATE_COMPONENT** (std::shared_ptr< oatpp::data::mapping::ObjectMapper >, apiObjectMapper)([]

Public Attributes

- **DatabaseComponent databaseComponent**
- **SwaggerComponent swaggerComponent**

Detailed Description

Class which creates and holds Application components and registers components in oatpp::base::Environment Order of components initialization is from top to bottom

Definition at line **23** of file **AppComponent.hpp**.

Member Function Documentation

primus::component::AppComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< oatpp::data::mapping::ObjectMapper > , apiObjectMapper) [inline]

Definition at line **52** of file **AppComponent.hpp**.

primus::component::AppComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< oatpp::network::ConnectionHandler > , serverConnectionHandler) [inline]

Definition at line **45** of file **AppComponent.hpp**.

primus::component::AppComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< oatpp::network::ServerConnectionProvider > , serverConnectionProvider) [inline]

Definition at line **33** of file **AppComponent.hpp**.

primus::component::AppComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< oatpp::web::server::HttpRouter > , httpRouter) [inline]

Definition at line **39** of file **AppComponent.hpp**.

Member Data Documentation

DatabaseComponent primus::component::AppComponent::databaseComponent

Definition at line **27** of file **AppComponent.hpp**.

SwaggerComponent primus::component::AppComponent::swaggerComponent

Definition at line **30** of file **AppComponent.hpp**.

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

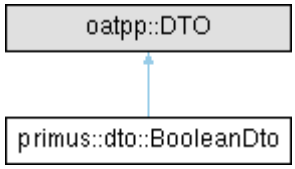
- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/AppComponent.hpp

primus::dto::BooleanDto Class Reference

DTO class representing a boolean value.

`#include <BooleanDto.hpp>`

Inheritance diagram for primus::dto::BooleanDto:



Detailed Description

DTO class representing a boolean value.

Definition at line **21** of file **BooleanDto.hpp**.

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

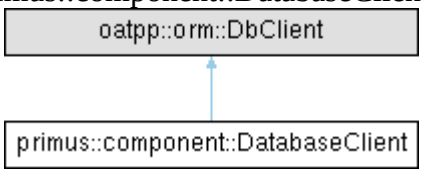
- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/BooleanDto.hpp

primus::component::DatabaseClient Class Reference

DatabaseClient class represents a client to interact with the database. End code-gen section.

`#include <DatabaseClient.hpp>`

Inheritance diagram for primus::component::DatabaseClient:



Public Member Functions

- **DatabaseClient** (const std::shared_ptr< oatpp::orm::Executor > &executor)
- **QUERY** (getMemberById, "SELECT * from Member WHERE id = :id;", PARAM(oatpp::UInt32, id))
- **QUERY** (createMember, "INSERT INTO Member (firstName, lastName, email, phoneNumber, birthDate, createDate, notes, active) " "SELECT :member.firstName, :member.lastName, :member.email, :member.phoneNumber, :member.birthDate, DATE('now'), :member.notes, :member.active " "WHERE NOT EXISTS (SELECT 1 FROM Member WHERE firstName = :member.firstName AND lastName = :member.lastName AND email = :member.email AND birthDate = :member.birthDate);", PARAM(oatpp::Object< **MemberDto** >, member))
- **QUERY** (updateMember, "UPDATE Member SET " "firstName = :member.firstName, " "lastName = :member.lastName, " "email = :member.email, " "phoneNumber = :member.phoneNumber, " "birthDate = :member.birthDate, " "notes = :member.notes, " "active = :member.active " "WHERE id = :member.id;", PARAM(oatpp::Object< **MemberDto** >, member))
- **QUERY** (findMemberIdByDetails, "SELECT * FROM Member WHERE firstName = :firstName AND lastName = :lastName AND email = :email AND birthDate = :birthDate;", PARAM(oatpp::String, firstName), PARAM(oatpp::String, lastName), PARAM(oatpp::String, email), PARAM(oatpp::String, birthDate))
- **QUERY** (activateMember, "UPDATE Member SET active = 1 WHERE id = :id;", PARAM(oatpp::UInt32, id))
- **QUERY** (deactivateMember, "UPDATE Member SET active = 0 WHERE id = :id;", PARAM(oatpp::UInt32, id))
- **QUERY** (getMemberCountAll, "SELECT COUNT(*) as value FROM Member")
- **QUERY** (getMemberCountActive, "SELECT COUNT(*) as value FROM Member WHERE active=1")
- **QUERY** (getMemberCountInactive, "SELECT COUNT(*) as value FROM Member WHERE active=0")
- **QUERY** (getMembersWithUpcomingBirthday, " SELECT * from Member m " " WHERE active = 1 AND strftime('%m-%d', m.birthDate) >= strftime('%m-%d', 'now') " " ORDER BY strftime('%m-%d', m.birthDate) ASC " " LIMIT :limit OFFSET :offset; ", PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (getAllMembers, " SELECT * FROM Member " " LIMIT :limit OFFSET :offset;", PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (getActiveMembers, " SELECT * FROM Member " " WHERE active = 1 " " ORDER BY id " " LIMIT :limit OFFSET :offset;", PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (getMembersByMostTraining, " SELECT m.* " " FROM Member m " " JOIN(" " SELECT member_id, COUNT(*) AS attendance_count " " FROM Attendance" " WHERE date >= date('now', '-6 months') " " GROUP BY member_id " " ORDER BY attendance_count DESC " " LIMIT :limit OFFSET :offset " ") AS top_members ON m.id = top_members.member_id; ", PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (getInactiveMembers, " SELECT * FROM Member " " WHERE active = 0 " " ORDER BY id " " LIMIT :limit OFFSET :offset;", PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))

- **QUERY** (getMembersByAddress, "SELECT Member.* FROM Member INNER JOIN Address_Member ON Member.id = Address_Member.member_id WHERE Address_Member.address_id = :addressId;", PARAM(oatpp::UInt32, addressId))
- **QUERY** (getMembersByDepartment, "SELECT Member.* FROM Member INNER JOIN Department_Member ON Member.id = Department_Member.member_id WHERE Department_Member.department_id = :departmentId;", PARAM(oatpp::UInt32, departmentId))
- **QUERY** (getDepartmentsOfMember, "SELECT d.* FROM Department d " "JOIN MemberDepartmentRel mdr ON d.id = mdr.departmentID " "WHERE mdr.memberID = :id;", PARAM(oatpp::UInt32, id))
- **QUERY** (getMemberAddresses, " SELECT a.* FROM Address a " " INNER JOIN Address_Member am ON a.id = am.address_id " " WHERE am.member_id = :id " " ORDER BY a.id " " LIMIT :limit OFFSET :offset;", PARAM(oatpp::UInt32, id), PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (getMemberDepartments, " SELECT d.* FROM Department d " " INNER JOIN Department_Member dm ON d.id = dm.department_id " " WHERE dm.member_id = :id " " ORDER BY d.id ASC" " LIMIT :limit OFFSET :offset;", PARAM(oatpp::UInt32, id), PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (getDepartmentById, " SELECT * FROM Department " " WHERE id = :id; " " ORDER BY id ASC", PARAM(oatpp::UInt32, id))
- **QUERY** (createAddress, " INSERT INTO Address (postalCode, city, country, houseNumber, street) " " SELECT :address.postalCode, :address.city, :address.country, :address.houseNumber, :address.street " " WHERE NOT EXISTS (SELECT 1 FROM Address WHERE postalCode = :address.postalCode AND city = :address.city AND country = :address.country AND houseNumber = :address.houseNumber AND street = :address.street);", PARAM(oatpp::Object< **AddressDto** >, address))
- **QUERY** (getAddressById, "SELECT * FROM Address WHERE id = :id;", PARAM(oatpp::UInt32, id))
- **QUERY** (updateAddress, "UPDATE Address SET " "street = :address.street, " "city = :address.city, " "zipCode = :address.zipCode, " "country = :address.country " "WHERE id = :address.id;", PARAM(oatpp::Object< **AddressDto** >, address))
- **QUERY** (deleteAddress, "DELETE FROM Address WHERE id = :id AND id NOT IN (SELECT address_id FROM Address_Member);", PARAM(oatpp::UInt32, id))
- **QUERY** (findAddressByDetails, "SELECT * FROM Address WHERE street = :street AND city = :city AND postalCode = :postalCode AND country = :country;", PARAM(oatpp::String, street), PARAM(oatpp::String, city), PARAM(oatpp::String, postalCode), PARAM(oatpp::String, country))
- **QUERY** (createMemberAttendance, "INSERT INTO Attendance (member_id, date) " "VALUES (:member_id, :date); ", PARAM(oatpp::UInt32, member_id), PARAM(oatpp::String, date))
- **QUERY** (deleteMemberAttendance, "DELETE FROM Attendance " "WHERE member_id = :member_id AND date = :date;", PARAM(oatpp::UInt32, member_id), PARAM(oatpp::String, date))
- **QUERY** (getAttendancesOfMember, " SELECT date FROM Attendance " " WHERE member_id = :member_id " " ORDER BY date DESC " " LIMIT :limit OFFSET :offset;", PARAM(oatpp::UInt32, member_id), PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (getCountOfAttendancesOfMember, " SELECT COUNT(*) as value FROM Attendance " " WHERE member_id = :member_id ", PARAM(oatpp::UInt32, member_id))
- **QUERY** (getMembersByAttendanceDate, " SELECT member_id as value FROM Attendance " " WHERE date = :date " " ORDER BY date " " LIMIT :limit OFFSET :offset;", PARAM(oatpp::String, dateOfAttendance), PARAM(oatpp::UInt32, limit), PARAM(oatpp::UInt32, offset))
- **QUERY** (hasMemberAttendedOnDate, " SELECT COUNT(*) as value FROM Attendance " " WHERE date = :date AND member_id = :member_id;", PARAM(oatpp::UInt32, member_id), PARAM(oatpp::String, date))
- **QUERY** (associateAddressWithMember, "INSERT INTO Address_Member (address_id, member_id) VALUES (:addressId, :memberId);", PARAM(oatpp::UInt32, addressId), PARAM(oatpp::UInt32, memberId))

- **QUERY** (disassociateAddressFromMember, "DELETE FROM Address_Member WHERE address_id = :addressId AND member_id = :memberId;", PARAM(oatpp::UInt32, addressId), PARAM(oatpp::UInt32, memberId))
- **QUERY** (associateDepartmentWithMember, "INSERT INTO Department_Member (department_id, member_id) VALUES (:departmentId, :memberId);", PARAM(oatpp::UInt32, departmentId), PARAM(oatpp::UInt32, memberId))
- **QUERY** (disassociateDepartmentFromMember, "DELETE FROM Department_Member WHERE department_id = :departmentId AND member_id = :memberId;", PARAM(oatpp::UInt32, departmentId), PARAM(oatpp::UInt32, memberId))
- **QUERY** (getCountOfMemberAttendancesInLastYear, " SELECT COUNT(*) as value " " FROM Attendance WHERE member_id = :memberId AND date >= DATE('now', '-1 year');", PARAM(oatpp::UInt32, memberId))
- **QUERY** (countDistinctAttendentMontsWithinLastYear, " SELECT COUNT(DISTINCT strftime('%m', date)) AS value " " FROM Attendance " " WHERE date >= DATE('now', '-1 year') " " AND member_id = :memberId; ", PARAM(oatpp::UInt32, memberId))

Detailed Description

DatabaseClient class represents a client to interact with the database. End code-gen section.

Definition at line **25** of file **DatabaseClient.hpp**.

Constructor & Destructor Documentation

primus::component::DatabaseClient::DatabaseClient (const std::shared_ptr< oatpp::orm::Executor > & *executor*) `[inline]`

Constructor to initialize the **DatabaseClient**.

Parameters

executor	- shared pointer to oatpp executor.

Definition at line **35** of file **DatabaseClient.hpp**.

Member Function Documentation

primus::component::DatabaseClient::QUERY (createMember , "INSERT INTO Member (firstName, lastName, email, phoneNumber, birthDate, createDate, notes, active) " "SELECT :member. *firstName*, :member. *lastName*, :member. *email*, :member. *phoneNumber*, :member. *birthDate*, DATE('now') , :member. *notes*, :member.active " "WHERE NOT EXISTS(SELECT 1 FROM Member WHERE firstName=:member.firstName AND lastName=:member.lastName AND email=:member.email AND birthDate=:member.birthDate);" , PARAM(oatpp::Object< MemberDto >, member))

Creates a member in the database

Parameters

member	A dto containing the mebers data

primus::component::DatabaseClient::QUERY (getMemberById)

Retrieves a member by id

Parameters

id	The member id

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/database/DatabaseClient.hpp

primus::component::DatabaseComponent Class Reference

Database component responsible for creating database connections and clients.
`#include <DatabaseComponent.hpp>`

Public Member Functions

- **OATPP_CREATE_COMPONENT** (std::shared_ptr< oatpp::provider::Provider< oatpp::sqlite::Connection > >, dbConnectionProvider)([]
- **OATPP_CREATE_COMPONENT** (std::shared_ptr< **DatabaseClient** >, database)([]

Detailed Description

Database component responsible for creating database connections and clients.

Definition at line **22** of file **DatabaseComponent.hpp**.

Member Function Documentation

primus::component::DatabaseComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< DatabaseClient > , database) `[inline]`

Definition at line **38** of file **DatabaseComponent.hpp**.

primus::component::DatabaseComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< oatpp::provider::Provider< oatpp::sqlite::Connection > > , dbConnectionProvider) `[inline]`

Definition at line **25** of file **DatabaseComponent.hpp**.

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

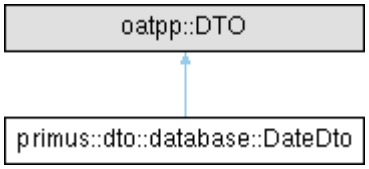
- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/database/DatabaseComponent.hpp

primus::dto::database::DateDto Class Reference

DTO class representing a single Date.

`#include <DatabaseDtos.hpp>`

Inheritance diagram for primus::dto::database::DateDto:



Detailed Description

DTO class representing a single Date.

Definition at line **67** of file **DatabaseDtos.hpp**.

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

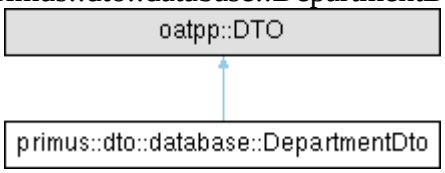
- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/DatabaseDtos.hpp

primus::dto::database::DepartmentDto Class Reference

DTO class representing a department.

`#include <DatabaseDtos.hpp>`

Inheritance diagram for primus::dto::database::DepartmentDto:



Detailed Description

DTO class representing a department.

Definition at line **84** of file **DatabaseDtos.hpp**.

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

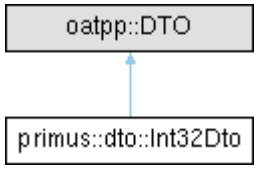
- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/DatabaseDtos.hpp

primus::dto::Int32Dto Class Reference

DTO class representing single value.

`#include <Int32Dto.hpp>`

Inheritance diagram for primus::dto::Int32Dto:



Detailed Description

DTO class representing single value.

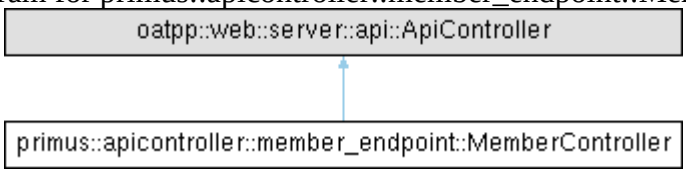
Definition at line **40** of file **Int32Dto.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/Int32Dto.hpp

primus::apicontroller::member_endpoint::MemberController Class Reference

Inheritance diagram for primus::apicontroller::member_endpoint::MemberController:



Public Member Functions

- **MemberController** (OATPP_COMPONENT(std::shared_ptr< ObjectMapper >, objectMapper))
- **ENDPOINT** ("GET", "/api/members/list/{attribute}", getMembersList, PATH(oatpp::String, attribute), QUERY(oatpp::UInt32, limit), QUERY(oatpp::UInt32, offset))
- **ENDPOINT** ("UPDATE", "/api/member/{id}/activate", activateMember, PATH(oatpp::UInt32, id))
- **ENDPOINT** ("UPDATE", "/api/member/{id}/deactivate", deactivateMember, PATH(oatpp::UInt32, id))
- **ENDPOINT** ("GET", "/api/member/{id}", getMemberById, PATH(oatpp::UInt32, id))
- **ENDPOINT** ("POST", "/api/member", createMember, BODY_DTO(Object< **MemberDto** >, member))
- **ENDPOINT** ("PUT", "/api/member", updateMember, BODY_DTO(Object< **MemberDto** >, member))
- **if** (attribute==oatpp::String("all"))
- else **if** (attribute==oatpp::String("active"))
- else **if** (attribute==oatpp::String("inactive"))
- **OATPP_LOGI** (primus::constants::apicontroller::member_endpoint::logName, "Returning CODE 500: Bad Request. Available options: all, active, inactive")
- return **createDtoResponse** (Status::CODE_404, status)
- **OATPP_LOGI** (primus::constants::apicontroller::member_endpoint::logName, "Received request to get count of %s members", attribute->c_str())
- **OATPP_ASSERT_HTTP** (dbResult->isSuccess(), Status::CODE_500, dbResult->getErrorMessage())
- **OATPP_LOGI** (primus::constants::apicontroller::member_endpoint::logName, "Processed request to get count of %s members. Total count: %d", attribute->c_str(), count[0]->value.operator v_uint32())
- return **createDtoResponse** (Status::CODE_200, count[0])
- **ENDPOINT** ("GET", "/api/member/{memberId}/count/{attribute}", getCountOfAttributeForMember, PATH(oatpp::UInt32, memberId), PATH(oatpp::String, attribute))
- **ENDPOINT** ("POST", "/api/member/{memberId}/department/add/{departmentId}", createMemberDepartmentAssociation, PATH(oatpp::UInt32, memberId), PATH(oatpp::UInt32, departmentId))
- **ENDPOINT** ("DELETE", "/api/member/{memberId}/department/remove/{departmentId}", deleteMemberDepartmentDisassociation, PATH(oatpp::UInt32, memberId), PATH(oatpp::UInt32, departmentId))
- **ENDPOINT** ("POST", "/api/member/{memberId}/address/add", createMemberAddressAssociation, PATH(oatpp::UInt32, memberId), BODY_DTO(oatpp::Object< **AddressDto** >, address))
- **ENDPOINT** ("DELETE", "/api/member/{memberId}/address/remove/{addressId}", deleteMemberAddressDisassociation, PATH(oatpp::UInt32, memberId), PATH(oatpp::UInt32, addressId))
- **ENDPOINT** ("POST", "/api/member/{memberId}/attendance/{dateOfAttendance}", addMemberAttendance, PATH(oatpp::UInt32, memberId), PATH(oatpp::String, dateOfAttendance))

- **ENDPOINT** ("DELETE", "/api/member/{memberId}/attendance/{dateOfAttendance}", deleteMemberAttendance, PATH(oatpp::UInt32, memberId), PATH(oatpp::String, dateOfAttendance))
- **ENDPOINT** ("GET", "/api/member/{memberId}/fee", getMemberFee, PATH(oatpp::UInt32, memberId))
- **ENDPOINT** ("GET", "/api/member/{memberId}/list/{attribute}", getMemberList, PATH(oatpp::UInt32, memberId), PATH(oatpp::String, attribute), QUERY(oatpp::UInt32, limit), QUERY(oatpp::UInt32, offset))
- **ENDPOINT** ("GET", "api/member/{memberId}/weaponpurchase/", canMemberBuyWeapon, PATH(oatpp::UInt32, memberId))
- **ENDPOINT_INFO** (getMembersList)
- **ENDPOINT_INFO** (activateMember)
- **ENDPOINT_INFO** (deactivateMember)
- **ENDPOINT_INFO** (getMemberById)
- **ENDPOINT_INFO** (createMember)
- **ENDPOINT_INFO** (updateMember)
- **ENDPOINT_INFO** (getMemberCount)
- **ENDPOINT_INFO** (createMemberDepartmentAssociation)
- **ENDPOINT_INFO** (deleteMemberDepartmentDisassociation)
- **ENDPOINT_INFO** (createMemberAddressAssociation)
- **ENDPOINT_INFO** (deleteMemberAddressDisassociation)
- **ENDPOINT_INFO** (addMemberAttendance)
- **ENDPOINT_INFO** (deleteMemberAttendance)
- **ENDPOINT_INFO** (getMemberFee)
- **ENDPOINT_INFO** (getMemberList)
- **ENDPOINT_INFO** (canMemberBuyWeapon)

Static Public Member Functions

- static std::shared_ptr< **MemberController** > **createShared** ()

Public Attributes

- **else**auto **status** = primus::dto::StatusDto::createShared()
- status **code** = 404
- status **message** = "Received request for a count of members with invalid attribute. Available options: all, active, inactive"
- status **status** = "INVALID ATTRIBUTE"
- auto **count** = dbResult->fetch<oatpp::Vector<oatpp::Object<**UInt32Dto**>>>()

Detailed Description

Definition at line **28** of file **MemberController.hpp**.

Constructor & Destructor Documentation

primus::apicontroller::member_endpoint::MemberController::MemberController (OATPP_COMPONENT(std::shared_ptr< ObjectMapper >, objectMapper)) `[inline]`

Definition at line **44** of file **MemberController.hpp**.

Member Function Documentation

static std::shared_ptr< MemberController > primus::apicontroller::member_endpoint::MemberController::createShared () `[inline]`, `[static]`

Definition at line **52** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("DELETE" , "/api/member/{memberId}/address/remove/{addressId}" , deleteMemberAddressDisassociation , PATH(oatpp::UInt32, memberId) , PATH(oatpp::UInt32, addressId)) `[inline]`

Definition at line **530** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("DELETE" , "/api/member/{memberId}/attendance/{dateOfAttendance}" , deleteMemberAttendance , PATH(oatpp::UInt32, memberId) , PATH(oatpp::String, dateOfAttendance)) `[inline]`

Definition at line **632** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("DELETE" , "/api/member/{memberId}/department/remove/{departmentId}" , deleteMemberDepartmentDisassociation , PATH(oatpp::UInt32, memberId) , PATH(oatpp::UInt32, departmentId)) `[inline]`

Definition at line **445** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("GET" , "/api/member/{id}" , getMemberById , PATH(oatpp::UInt32, id)) `[inline]`

Definition at line **176** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("GET" , "/api/member/{memberId}/count/{attribute}" , getCountOfAttributeForMember , PATH(oatpp::UInt32, memberId) , PATH(oatpp::String, attribute)) `[inline]`

Definition at line **373** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("GET" , "/api/member/{memberId}/fee" , getMemberFee , PATH(oatpp::UInt32, memberId)) `[inline]`

Definition at line **659** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("GET" , "/api/member/{memberId}/list/{attribute}" , getMemberList , PATH(oatpp::UInt32, memberId) , PATH(oatpp::String, attribute) , QUERY(oatpp::UInt32, limit) , QUERY(oatpp::UInt32, offset)) `[inline]`

Definition at line **718** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("GET" , "/api/members/list/{attribute}" , getMembersList , PATH(oatpp::String, attribute) , QUERY(oatpp::UInt32, limit) , QUERY(oatpp::UInt32, offset)) `[inline]`

Definition at line **59** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("GET" , "api/member/{memberId}/weaponpurchase/" , canMemberBuyWeapon , PATH(oatpp::UInt32, memberId)) `[inline]`

Definition at line **813** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("POST" , "/api/member" , createMember , BODY_DTO(Object< MemberDto >, member)) `[inline]`

Definition at line **202** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("POST" , "/api/member/{memberId}/address/add" , createMemberAddressAssociation , PATH(oatpp::UInt32, memberId) , BODY_DTO(oatpp::Object< AddressDto >, address)) `[inline]`

Definition at line **481** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("POST" , "/api/member/{memberId}/attendance/{dateOfAttendance}" , addMemberAttendance , PATH(oatpp::UInt32, memberId) , PATH(oatpp::String, dateOfAttendance)) `[inline]`

Definition at line **589** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("POST" , "/api/member/{memberId}/department/add/{departmentId}" , createMemberDepartmentAssociation , PATH(oatpp::UInt32, memberId) , PATH(oatpp::UInt32, departmentId)) `[inline]`

Definition at line **400** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("PUT" , "/api/member" , updateMember , BODY_DTO(Object< MemberDto >, member)) `[inline]`

Definition at line **261** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("UPDATE" , "/api/member/{id}/activate" , activateMember , PATH(oatpp::UInt32, id)) `[inline]`

Definition at line **121** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT ("UPDATE" , "/api/member/{id}/deactivate" , deactivateMember , PATH(oatpp::UInt32, id)) `[inline]`

Definition at line **149** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (activateMember) `[inline]`

Definition at line **890** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (addMemberAttendance) `[inline]`

Definition at line **1029** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (canMemberBuyWeapon) `[inline]`

Definition at line **1096** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (createMember) `[inline]`

Definition at line **927** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (createMemberAddressAssociation) `[inline]`

Definition at line **1000** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (createMemberDepartmentAssociation) `[inline]`

Definition at line **968** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (deactivateMember) `[inline]`

Definition at line **902** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (deleteMemberAddressDisassociation) `[inline]`

Definition at line **1014** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (deleteMemberAttendance) `[inline]`

Definition at line **1045** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (deleteMemberDepartmentDisassociation) `[inline]`

Definition at line **984** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (getMemberById) `[inline]`

Definition at line **914** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (getMemberCount) `[inline]`

Definition at line **953** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (getMemberFee) `[inline]`

Definition at line **1061** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (getMemberList) `[inline]`

Definition at line **1076** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (getMembersList) `[inline]`

Definition at line **873** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::ENDPOINT_INFO (updateMember) `[inline]`

Definition at line **940** of file **MemberController.hpp**.

else primus::apicontroller::member_endpoint::MemberController::if (attribute == oatpp::String("active")) `[inline]`

Definition at line **343** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::if (attribute == oatpp::String("all")) `[inline]`

Definition at line **339** of file **MemberController.hpp**.

else primus::apicontroller::member_endpoint::MemberController::if (attribute == oatpp::String("inactive")) `[inline]`

Definition at line **347** of file **MemberController.hpp**.

Member Data Documentation

status primus::apicontroller::member_endpoint::MemberController::code = 404

Definition at line **357** of file **MemberController.hpp**.

auto primus::apicontroller::member_endpoint::MemberController::count = dbResult->fetch<oatpp::Vector<oatpp::Object<UInt32Dto>>>()

Definition at line **366** of file **MemberController.hpp**.

primus::apicontroller::member_endpoint::MemberController::else

```
Initial value:{

OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
count request for members with attribute %s, which is not an available attribute",
attribute->c_str())
```

Definition at line **351** of file **MemberController.hpp**.

status primus::apicontroller::member_endpoint::MemberController::message = "Received request for a count of members with invalid attribute. Available options: all, active, inactive"

Definition at line **358** of file **MemberController.hpp**.

auto primus::apicontroller::member_endpoint::MemberController::status = primus::dto::StatusDto::createShared()

Definition at line **356** of file **MemberController.hpp**.

status primus::apicontroller::member_endpoint::MemberController::status = "INVALID ATTRIBUTE"

Definition at line **359** of file **MemberController.hpp**.

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

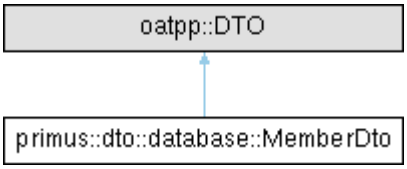
- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/controller/MemberController.hpp

primus::dto::database::MemberDto Class Reference

DTO class representing a member.

`#include <DatabaseDtos.hpp>`

Inheritance diagram for primus::dto::database::MemberDto:



Detailed Description

DTO class representing a member.

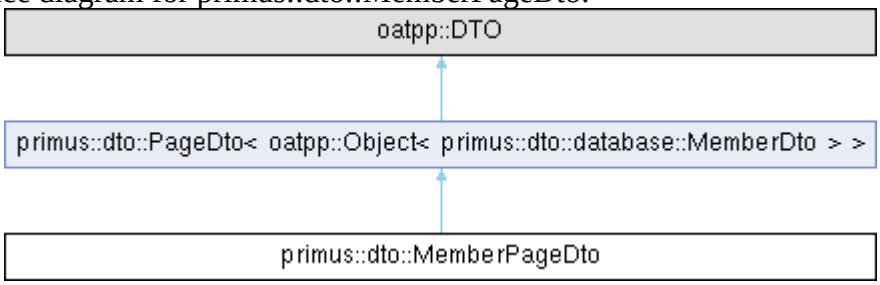
Definition at line **109** of file **DatabaseDtos.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/DatabaseDtos.hpp

primus::dto::MemberPageDto Class Reference

Inheritance diagram for primus::dto::MemberPageDto:



Detailed Description

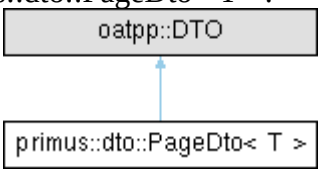
Definition at line **51** of file **PageDto.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/PageDto.hpp

primus::dto::PageDto< T > Class Template Reference

Inheritance diagram for primus::dto::PageDto< T >:



Detailed Description

template<class T>

class primus::dto::PageDto< T >

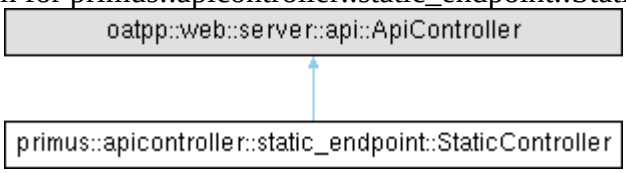
Definition at line **20** of file **PageDto.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/PageDto.hpp

primus::apicontroller::static_endpoint::StaticController Class Reference

Inheritance diagram for primus::apicontroller::static_endpoint::StaticController:



Public Member Functions

- **StaticController** (OATPP_COMPONENT(std::shared_ptr< ObjectMapper >, objectMapper))
- **ENDPOINT** ("GET", "/web/*", files, REQUEST(std::shared_ptr< IncomingRequest >, request))
- **ENDPOINT** ("GET", "/", root, REQUEST(std::shared_ptr< IncomingRequest >, request))
- std::string **filePath** (USER_ASSETS)
- filePath **append** ("/")
- **if** (userStatus->code !=200)
- filePath **append** (memberId)
- filePath **append** (".jpg")
- **OATPP_LOGI** (primus::constants::apicontroller::static_endpoint::logName, "Serving file: %s", filePath.c_str())
- std::ifstream **file** (filePath, std::ios::binary)
- **if** (!file.good())
- **OATPP_LOGI** (primus::constants::apicontroller::static_endpoint::logName, "File at %s found", finalPath.c_str())
- file **close** ()
- return **createResponse** (Status::CODE_200, content.str())
- **ENDPOINT_INFO** (getAvatar)
- **ENDPOINT_INFO** (files)
- **ENDPOINT_INFO** (root)

Static Public Member Functions

- static std::shared_ptr< **StaticController** > **createShared** ()

Public Attributes

- std::string **finalPath**
- std::ostringstream **content**
- int **choice**
- auto **since_epoch** = now.time_since_epoch()
- auto **millis** = std::chrono::duration_cast<std::chrono::milliseconds>(since_epoch).count()
- **choice** = millis % 2
- auto **userStatus** = primus::assert::assertMemberExists(oatpp::utils::conversion::strToUInt32(memberId->c_str()))
- **else**
- **finalPath** = filePath

Detailed Description

Definition at line **22** of file **StaticController.hpp**.

Constructor & Destructor Documentation

primus::apicontroller::static_endpoint::StaticController::StaticController (OATPP_COMPONENT(std::shared_ptr< ObjectMapper >, objectMapper)) `[inline]`

Definition at line **25** of file **StaticController.hpp**.

Member Function Documentation

static std::shared_ptr< StaticController > primus::apicontroller::static_endpoint::StaticController::createShared () `[inline], [static]`

Definition at line **35** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::ENDPOINT ("GET" , "/" , root , REQUEST(std::shared_ptr< IncomingRequest >, request)) `[inline]`

Definition at line **95** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::ENDPOINT ("GET" , "/web/*" , files , REQUEST(std::shared_ptr< IncomingRequest >, request)) `[inline]`

Definition at line **42** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::ENDPOINT_INFO (files) `[inline]`

Definition at line **206** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::ENDPOINT_INFO (getAvatar) `[inline]`

Definition at line **195** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::ENDPOINT_INFO (root) `[inline]`

Definition at line **219** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::if (!file. *good*()) `[inline]`

Definition at line **148** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::if (userStatus->code ! = 200) `[inline]`

Definition at line **129** of file **StaticController.hpp**.

Member Data Documentation

int primus::apicontroller::static_endpoint::StaticController::choice

Definition at line **118** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::choice = millis % 2

Definition at line **124** of file **StaticController.hpp**.

std::ostringstream primus::apicontroller::static_endpoint::StaticController::content

Definition at line **117** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::else

```
Initial value:{

OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "User found.
Looking for profile picture within directory")
```

Definition at line **136** of file **StaticController.hpp**.

std::string primus::apicontroller::static_endpoint::StaticController::finalPath

Definition at line **116** of file **StaticController.hpp**.

primus::apicontroller::static_endpoint::StaticController::finalPath = filePath

Definition at line **143** of file **StaticController.hpp**.

auto primus::apicontroller::static_endpoint::StaticController::millis = std::chrono::duration_cast<std::chrono::milliseconds>(since_epoch).count()

Definition at line **123** of file **StaticController.hpp**.

auto primus::apicontroller::static_endpoint::StaticController::since_epoch = now.time_since_epoch()

Definition at line **122** of file **StaticController.hpp**.

auto primus::apicontroller::static_endpoint::StaticController::userStatus = primus::assert::assertMemberExists(oatpp::utils::conversion::strToUInt32(memberId->c_str()))

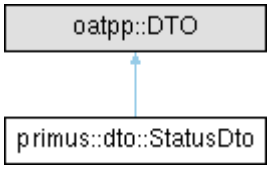
Definition at line **127** of file **StaticController.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/controller/StaticController.hpp

primus::dto::StatusDto Class Reference

Inheritance diagram for primus::dto::StatusDto:



Detailed Description

Definition at line **17** of file **StatusDto.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/StatusDto.hpp

primus::component::SwaggerComponent Class Reference

`#include <SwaggerComponent.hpp>`

Public Member Functions

- **OATPP_CREATE_COMPONENT** (std::shared_ptr< oatpp::swagger::DocumentInfo >, swaggerDocumentInfo)([]
- **OATPP_CREATE_COMPONENT** (std::shared_ptr< oatpp::swagger::Resources >, swaggerResources)([]

Detailed Description

Swagger ui is served at `http://host:port/swagger/ui`

Definition at line **17** of file **SwaggerComponent.hpp**.

Member Function Documentation

primus::component::SwaggerComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< oatpp::swagger::DocumentInfo > , swaggerDocumentInfo) `[inline]`

General API docs info

Definition at line **23** of file **SwaggerComponent.hpp**.

primus::component::SwaggerComponent::OATPP_CREATE_COMPONENT (std::shared_ptr< oatpp::swagger::Resources > , swaggerResources) `[inline]`

Swagger-Ui Resources (<oatpp-examples>/lib/oatpp-swagger/res)

Definition at line **46** of file **SwaggerComponent.hpp**.

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

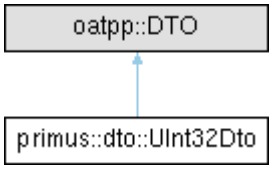
- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/swagger-ui/SwaggerComponent.hpp

primus::dto::UInt32Dto Class Reference

DTO class representing single value.

`#include <Int32Dto.hpp>`

Inheritance diagram for primus::dto::UInt32Dto:



Detailed Description

DTO class representing single value.

Definition at line **21** of file **Int32Dto.hpp**.

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

- F:/Programming/School/repos/BSFI22D_Primus/PrimusSvr/src/dto/Int32Dto.hpp

File Documentation

App.cpp

```
00001 #include "AppComponent.hpp"
00002
00003 // Controller includes
00004 #include "general/asserts.hpp"
00005 #include "controller/StaticController.hpp"
00006 #include "controller/MemberController.hpp"
00007 #include "oatpp-swagger/Controller.hpp"
00008 #include "oatpp/network/Server.hpp"
00009 #include <iostream>
00010
00011 // __| |_____________________________________| |__
00012 // __   _____________________________________   __
00013 //   | |                                     | |
00014 //   | |  ____          _                    | |
00015 //   | | |  _ \ _ __(_)_ __ ___  _   _ ___  | |
00016 //   | | | |_) | '__| | '_ ` _ \| | | / __| | |
00017 //   | | |  __/| |  | | | | | | | |_| \__ \| |
00018 //   | |_|   |_|  |_|_| |_| |_|\__,_|___/| |
00019 // __| |_____________________________________| |__
00020 // __   _____________________________________   __
00021 //   | |                                     | |
00022 namespace primus {
00023     namespace main {
00024         void run(void) {
00025             typedef primus::apicontroller::static_endpoint::StaticController
StaticController;
00026             typedef primus::apicontroller::member_endpoint::MemberController
MemberController;
00027             typedef primus::component::AppComponent
AppComponent;
00028             typedef primus::component::DatabaseClient
DatabaseClient;
00029             typedef primus::component::DatabaseComponent
DatabaseComponent;
00030             typedef primus::component::SwaggerComponent
SwaggerComponent;
00031
00032             OATPP_LOGI(primus::constants::main::logName, "Initializing
AppComponents");
00033
00034             /* Register Components in scope of run() method */
00035             AppComponent components;
00036
00037             OATPP_LOGI(primus::constants::main::logName, "AppComponents have been
initialized");
00038             OATPP_LOGI(primus::constants::main::logName, "Creating additional
component router (oatpp::web::server::HttpRouter)");
00039
00040             /* Get router component */
00041             OATPP_COMPONENT(std::shared_ptr<oatpp::web::server::HttpRouter>,
router);
00042
00043             OATPP_LOGI(primus::constants::main::logName, "router
(oatpp::web::server::HttpRouter) has been initialized");
00044             OATPP_LOGI(primus::constants::main::logName, "Adding Endpoints of
StaticController to router");
00045
00046             /* Create StaticController and add all of its endpoints to router */
00047             router->addController(std::make_shared<StaticController>());
00048
00049             OATPP_LOGI(primus::constants::main::logName, "Endpoints of
StaticController successfully added");
00050             OATPP_LOGI(primus::constants::main::logName, "Adding Endpoints of
MemberController to router");
00051
00052             /* Create MemberController and add all of its endpoints to router */
00053             router->addController(std::make_shared<MemberController>());
00054
```

```
00055             OATPP_LOGI(primus::constants::main::logName, "Endpoints of
MemberController successfully added");
00056             OATPP_LOGI(primus::constants::main::logName, "Endpoints of
MemberController successfully added");
00057             OATPP_LOGI(primus::constants::main::logName, "Collecting Endpoints
for swagger-ui...");
00058
00059             /* Swagger UI Endpoint documentation */
00060             oatpp::web::server::api::Endpoints docEndpoints;
00061
00062
docEndpoints.append(router->addController(StaticController::createShared())->getEndpoi
nts());
00063             OATPP_LOGI(primus::constants::main::logName, "Collected Endpoints of
StaticController");
00064
00065
docEndpoints.append(router->addController(MemberController::createShared())->getEndpoi
nts());                     // Add the endpoints of MemberController to the swagger ui
documentation
00066             OATPP_LOGI(primus::constants::main::logName, "Collected Endpoints of
MemberController");
00067
00068             OATPP_LOGI(primus::constants::main::logName, "Initializing Swagger
Endpoint-Controller (oatpp::swagger::Controller) with collected endpoints");
00069
router->addController(oatpp::swagger::Controller::createShared(docEndpoints));
00070
00071             OATPP_LOGI(primus::constants::main::logName, "Creating additional
component connectionHandler (oatpp::network::ConnectionHandler)");
00072
00073             /* Get connection handler component */
00074             OATPP_COMPONENT(std::shared_ptr<oatpp::network::ConnectionHandler>,
connectionHandler);
00075
00076             OATPP_LOGI(primus::constants::main::logName,
"connectionHandler(oatpp::network::ConnectionHandler) successfully initialized");
00077
00078             OATPP_LOGI(primus::constants::main::logName, "Creating additional
component ServerConnectionProvider (oatpp::network::ServerConnectionProvider)");
00079
00080             /* Get connection provider component */
00081
OATPP_COMPONENT(std::shared_ptr<oatpp::network::ServerConnectionProvider>,
connectionProvider);
00082
00083             OATPP_LOGI(primus::constants::main::logName,
"ServerConnectionProvider (oatpp::network::ServerConnectionProvider) successfully
initialized");
00084             OATPP_LOGI(primus::constants::main::logName, "Initializing server
(oatpp::network::Server) with connectionProvider and connectionHandler");
00085
00086             /* Create server which takes provided TCP connections and passes them
to HTTP connection handler */
00087             oatpp::network::Server server(connectionProvider, connectionHandler);
00088
00089             OATPP_LOGI(primus::constants::main::logName, "Server has been
initialized");
00090
00091             OATPP_LOGI(primus::constants::main::logName, "Starting server");
00092             OATPP_LOGI(primus::constants::main::logName, "Host: %s",
connectionProvider->getProperty("host").getData());
00093             OATPP_LOGI(primus::constants::main::logName, "Port: %s",
connectionProvider->getProperty("port").getData());
00094
00095             OATPP_LOGI(primus::constants::main::logName, "__|
|                                  | |   ");
00096             OATPP_LOGI(primus::constants::main::logName, " __
_____________________________   __");
00097             OATPP_LOGI(primus::constants::main::logName, "   | |
| |  ");
00098             OATPP_LOGI(primus::constants::main::logName, "   | |
| |  ");
00099             OATPP_LOGI(primus::constants::main::logName, "   | ||  _ \\ _ __(_)_ __
___ _   _ ___ | |  ");
00100             OATPP_LOGI(primus::constants::main::logName, "   | || |_) | '__| | '_
` _ \\| | | / __|| |  ");
```

```
00101             OATPP_LOGI(primus::constants::main::logName, "   | ||  __/| |  | | | |
| | | |_| \\__ \\| |  ");
00102             OATPP_LOGI(primus::constants::main::logName, "   | ||_|   |_|  |_|_|
|_| |_|\\__,_|___/| |  ");
00103             OATPP_LOGI(primus::constants::main::logName, " __|
|_____________________________________| |__");
00104             OATPP_LOGI(primus::constants::main::logName, " __
_____________________________________   __");
00105             OATPP_LOGI(primus::constants::main::logName, "   | |
| |  ");
00106
00107             /* Run server */
00108             server.run();
00109
00110         } // run()
00111     } // namespace main
00112 } // namespace primus
00113
00114 //  __   __        _
00115 // |  \/  | __ _(_)_ __
00116 // | |\/| |/ _` | | '_ \
00117 // | |  | | (_| | | | | |
00118 // |_|  |_|\__,_|_|_| |_|
00122 int main(int argc, const char* argv[])
00123 {
00124     oatpp::base::Environment::init();
00125
00126     primus::main::run();
00127
00128     /* Print how much objects were created during app running, and what have
left-probably leaked */
00129     /* Disable object counting for release builds using '-D
OATPP_DISABLE_ENV_OBJECT_COUNTERS' flag for better performance */
00130     std::cout << "\nEnvironment:\n";
00131     std::cout << "objectsCount = " << oatpp::base::Environment::getObjectsCount()
<< "\n";
00132     std::cout << "objectsCreated = " <<
oatpp::base::Environment::getObjectsCreated() << "\n\n";
00133
00134     oatpp::base::Environment::destroy();
00135
00136     return 0;
00137 }
```

AppComponent.hpp

```
00001 #ifndef AppComponent_hpp
00002 #define AppComponent_hpp
00003
00004 // Oatpp headers
00005 #include "oatpp/web/server/HttpConnectionHandler.hpp"
00006 #include "oatpp/web/server/HttpRouter.hpp"
00007 #include "oatpp/network/tcp/server/ConnectionProvider.hpp"
00008 #include "oatpp/parser/json/mapping/ObjectMapper.hpp"
00009 #include "oatpp/core/macro/component.hpp"
00010
00011 // App specific headers
00012 #include "database/DatabaseComponent.hpp"
00013 #include "swagger-ui/SwaggerComponent.hpp"
00014
00015 namespace primus
00016 {
00017     namespace component
00018     {
00023         class AppComponent
00024         {
00025         public:
00026             // Database component
00027             DatabaseComponent databaseComponent;
00028
00029             // Swagger component
00030             SwaggerComponent swaggerComponent;
00031
00032             // Create ConnectionProvider component which listens on the port
00033
OATPP_CREATE_COMPONENT(std::shared_ptr<oatpp::network::ServerConnectionProvider>,
serverConnectionProvider)([] {
00034                 return
oatpp::network::tcp::server::ConnectionProvider::createShared({ "0.0.0.0", 8000,
oatpp::network::Address::IP_4 });
00035                 }());
00036
00037
00038             // Create Router component
00039
OATPP_CREATE_COMPONENT(std::shared_ptr<oatpp::web::server::HttpRouter>, httpRouter)([]
{
00040                 return oatpp::web::server::HttpRouter::createShared();
00041                 }());
00042
00043
00044             // Create ConnectionHandler component which uses Router component to
route requests
00045
OATPP_CREATE_COMPONENT(std::shared_ptr<oatpp::network::ConnectionHandler>,
serverConnectionHandler)([] {
00046                 OATPP_COMPONENT(std::shared_ptr<oatpp::web::server::HttpRouter>,
router); // get Router component
00047                 return
oatpp::web::server::HttpConnectionHandler::createShared(router);
00048                 }());
00049
00050
00051             // Create ObjectMapper component to serialize/deserialize Dtos in
Contoller's API
00052
OATPP_CREATE_COMPONENT(std::shared_ptr<oatpp::data::mapping::ObjectMapper>,
apiObjectMapper)([] {
00053                 return
oatpp::parser::json::mapping::ObjectMapper::createShared();
00054                 }());
00055         };
00056
00057     } //namespace component
00058 } // namespace primus
00059
00060 #endif /* AppComponent_hpp */
```

MemberController.hpp

```
00001 #ifndef MEMBERCONTROLLER_HPP
00002 #define MEMBERCONTROLLER_HPP
00003
00004 #include "oatpp/web/server/api/ApiController.hpp"
00005 #include "oatpp/core/macro/codegen.hpp"
00006 #include "oatpp/core/macro/component.hpp"
00007 #include "dto/StatusDto.hpp"
00008 #include "dto/PageDto.hpp"
00009 #include "dto/Int32Dto.hpp"
00010 #include "dto/BooleanDto.hpp"
00011 #include "general/constants.hpp"
00012 #include "assert.h"
00013
00014 namespace primus {
00015     namespace apicontroller {
00016         namespace member_endpoint {
00017
00018 #include OATPP_CODEGEN_BEGIN(ApiController) // Begin API Controller codegen
00019
00020
00021             using primus::dto::PageDto;
00022             //  __  __                  _                ____            _
__ __
00023             // |  \/  | ___ _ __ ___ | |__   ___ _ __ / ___|___  _ __ | |_ _ __ ___
| | | ___ _ __
00024             // | |\/| |/ _ \ '_ ` _ \| '_ \ / _ \ '__| |   / _ \| '_ \| __| '__/ _
\| | |/ _ \ '__|
00025             // | |  | |  __/ | | | | | |_) |  __/ |  | |__| (_) | | | | |_| | | (_)
| | |  __/ |
00026             // |_|  |_|\___|_| |_| |_|_.__/ \___|_|   \____\___/|_| |_|\__|_|
\___/|_|_|\___|_|
00027
00028             class MemberController : public oatpp::web::server::api::ApiController
00029             {
00030                 typedef primus::dto::database::MemberDto MemberDto;
00031                 typedef primus::dto::database::DepartmentDto DepartmentDto;
00032                 typedef primus::dto::database::AddressDto AddressDto;
00033                 typedef primus::dto::database::DateDto DateDto;
00034                 typedef primus::dto::MemberPageDto MemberPageDto;
00035                 typedef primus::dto::UInt32Dto UInt32Dto;
00036                 typedef primus::dto::Int32Dto Int32Dto;
00037                 typedef primus::dto::BooleanDto BooleanDto;
00038                 typedef primus::dto::StatusDto StatusDto;
00039
00040             private:
00041
OATPP_COMPONENT(std::shared_ptr<primus::component::DatabaseClient>, m_database);
00042
00043             public:
00044                 MemberController(OATPP_COMPONENT(std::shared_ptr<ObjectMapper>,
objectMapper))
00045                     : oatpp::web::server::api::ApiController(objectMapper)
00046                 {
00047
00048
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName,
"MemberController (oatpp::web::server::api::ApiController) initialized");
00049
00050                 }
00051
00052                 static std::shared_ptr<MemberController> createShared(
00053                     OATPP_COMPONENT(std::shared_ptr<ObjectMapper>, objectMapper)
00054                 )
00055                 {
00056                     return std::make_shared<MemberController>(objectMapper);
00057                 }
00058
00059                 ENDPOINT("GET", "/api/members/list/{attribute}", getMembersList,
00060                     PATH(oatpp::String, attribute), QUERY(oatpp::UInt32, limit),
QUERY(oatpp::UInt32, offset))
00061                 {
00062
00063                     std::shared_ptr<oatpp::orm::QueryResult> dbResult;
```

```
00064                     if (attribute == oatpp::String("all"))
00065                     {
00066
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list of all members. Limit: %d, Offset: %d", limit.operator v_uint32(),
offset.operator v_uint32());
00067
00068                         dbResult = m_database->getAllMembers(limit, offset);
00069                     }
00070                     else if (attribute == oatpp::String("active"))
00071                     {
00072
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list of all active members. Limit: %d, Offset: %d", limit.operator
v_uint32(), offset.operator v_uint32());
00073
00074                         dbResult = m_database->getActiveMembers(limit, offset);
00075                     }
00076                     else if (attribute == oatpp::String("inactive"))
00077                     {
00078
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list of all inactive members. Limit: %d, Offset: %d", limit.operator
v_uint32(), offset.operator v_uint32());
00079
00080                         dbResult = m_database->getInactiveMembers(limit, offset);
00081                     }
00082                     else if (attribute == oatpp::String("birthday"))
00083                     {
00084
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list of all members with upcomming birthdays. Limit: %d, Offset: %d",
limit.operator v_uint32(), offset.operator v_uint32());
00085
00086                         dbResult =
m_database->getMembersWithUpcomingBirthday(limit, offset);
00087                     }
00088                     else if (attribute == oatpp::String("training-most-often"))
00089                     {
00090
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list of all members with most training. Limit: %d, Offset: %d",
limit.operator v_uint32(), offset.operator v_uint32());
00091
00092                         dbResult = m_database->getMembersByMostTraining(limit,
offset);
00093                     }
00094                     else
00095                     {
00096
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list of members with %s, which is not an available attribute",
attribute->c_str());
00097
OATPP LOGI(primus::constants::apicontroller::member endpoint::logName, "Returning CODE
404: NOT FOUND. Available options: birthday, all, active or inactive");
00098
00099                         auto status = primus::dto::StatusDto::createShared();
00100                         status->code = 404;
00101                         status->message = "Received request to get a list of members
with invalid attribute. Available options: birthday, all, active or inactive";
00102                         status->status = "INVALID ATTRIBUTE";
00103                         return createDtoResponse(Status::CODE_200, status);
00104                     }
00105                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00106
00107                     auto items =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00108
00109                     auto page =
PageDto<oatpp::Object<MemberDto>>::createShared();
00110
00111                     page->offset = offset;
00112                     page->limit = limit;
00113                     page->count = items->size();
00114                     page->items = items;
```

```
00115
00116
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Processed
request to get a list of members with %s. Limit: %d, Offset: %d. Returned %d items",
attribute->c_str(), limit.operator v_uint32(), offset.operator v_uint32(),
page->count.operator v_uint32());
00117
00118                   return createDtoResponse(Status::CODE_200, page);
00119               }
00120
00121               ENDPOINT("UPDATE", "/api/member/{id}/activate", activateMember,
00122                   PATH(oatpp::UInt32, id))
00123               {
00124
00125
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to activate member with id: %d", id);
00126
00127                   {
00128                       auto status = primus::assert::assertMemberExists(id);
00129
00130                       if (status->code != 200)
00131                       {
00132                           return createDtoResponse(Status::CODE_500, status);
00133                       }
00134                   }
00135
00136                   auto dbResult = m_database->activateMember(id);
00137                   OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
"Unknown error");
00138
00139
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member with id:
%d activated", id);
00140
00141
00142                   auto status = primus::dto::StatusDto::createShared();
00143                   status->code = 200;
00144                   status->message = "Member has been activated";
00145                   status->status = "Member activated";
00146                   return createDtoResponse(Status::CODE_200, status);
00147               }
00148
00149               ENDPOINT("UPDATE", "/api/member/{id}/deactivate",
deactivateMember,
00150                   PATH(oatpp::UInt32, id))
00151               {
00152
00153
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to deactivate member with id: %d", id);
00154
00155                   {
00156                       auto status = primus::assert::assertMemberExists(id);
00157                       if (status->code != 200)
00158                       {
00159                           return createDtoResponse(Status::CODE_500, status);
00160                       }
00161                   }
00162
00163                   auto dbResult = m database->deactivateMember(id);
00164                   OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
"UNKNOWN ERROR");
00165
00166
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member with id:
%d deactivated", id);
00167
00168
00169                   auto status = primus::dto::StatusDto::createShared();
00170                   status->code = 200;
00171                   status->message = "Member has been deactivated";
00172                   status->status = "Member deactivated";
00173                   return createDtoResponse(Status::CODE_200, status);
00174               }
00175
00176               ENDPOINT("GET", "/api/member/{id}", getMemberById,
```

```
00177                       PATH(oatpp::UInt32, id))
00178                   {
00179
00180
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get member by id: %d", id.operator v_uint32());
00181
00182                       auto status = primus::assert::assertMemberExists(id);
00183
00184                       if (status->code != 200)
00185                       {
00186                           return createDtoResponse(Status::CODE_500, status);
00187                       }
00188
00189                       auto dbResult = m_database->getMemberById(id);
00190
00191                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00192                       OATPP_ASSERT_HTTP(dbResult->hasMoreToFetch(),
Status::CODE_404, "Member not found");
00193
00194                       auto result =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00195                       OATPP_ASSERT_HTTP(result->size() == 1, Status::CODE_500,
"Unknown error");
00196
00197
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Processed
request to get member by id: %d", id.operator v_uint32());
00198
00199                       return createDtoResponse(Status::CODE_200, result[0]);
00200                   }
00201
00202                   ENDPOINT("POST", "/api/member", createMember,
00203                       BODY_DTO(Object<MemberDto>, member))
00204                   {
00205
00206
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to create member");
00207
00208                       member->firstName    = member->firstName  == nullptr ? "" :
member->firstName ;
00209                       member->lastName     = member->lastName   == nullptr ? "" :
member->lastName  ;
00210                       member->email        = member->email      == nullptr ? "" :
member->email     ;
00211                       member->phoneNumber  = member->phoneNumber  == nullptr ? "" :
member->phoneNumber ;
00212                       member->birthDate    = member->birthDate  == nullptr ? "" :
member->birthDate ;
00213                       member->createDate   = member->createDate == nullptr ? "" :
member->createDate ;
00214                       member->notes        = member->notes      == nullptr ? "" :
member->notes     ;
00215                       member->active       = member->active     == nullptr ? "" :
member->active    ;
00216
00217
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member data:");
00218
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - First Name:
%s",    member->firstName->c_str());
00219
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Last Name:
%s",     member->lastName->c_str());
00220
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Email: %s",
member->email->c_str());
00221
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Phone
Number: %s",  member->phoneNumber->c str());
00222
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Birth Date:
%s",    member->birthDate->c_str());
```

```
00223
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Create
Date: %s",   member->createDate->c_str());
00224
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Notes: %s",
member->notes->c_str());
00225
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Active:
%s",        member->active ? "true" : "false");
00226
00227                     std::shared_ptr<oatpp::orm::QueryResult> dbResult =
m_database->createMember(member);
00228                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
"Bad Request");
00229
00230                     oatpp::UInt32 memberId =
oatpp::sqlite::Utils::getLastInsertRowId(dbResult->getConnection());
00231
00232                     oatpp::Vector<oatpp::Object<MemberDto>> foundMembers;
00233                     oatpp::Object<MemberDto> retMember;
00234
00235                     if (memberId == 0)
00236                     {
00237
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member already
exists. proceeding to return existing user");
00238
00239                         dbResult =
m_database->findMemberIdByDetails(member->firstName, member->lastName, member->email,
member->birthDate);
00240                         OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, "Unknown error");
00241
00242                         foundMembers =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00243
00244                         retMember = foundMembers[0];
00245                     }
00246                     else
00247                     {
00248
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Created member
with id: %d", memberId.operator v_uint32());
00249
00250                         dbResult = m_database->getMemberById(memberId);
00251                         OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, "Unknown error");
00252
00253                         foundMembers =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00254
00255                         retMember = foundMembers[0];
00256                     }
00257
00258                     return createDtoResponse(Status::CODE 200, retMember);
00259                 }
00260
00261                 ENDPOINT("PUT", "/api/member", updateMember,
00262                     BODY_DTO(Object<MemberDto>, member))
00263                 {
00264                         member->id = member->id == nullptr ? 0 : member->id;
00265
00266
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to update member with id: %d", member->id.operator v_uint32());
00267
00268                     {
00269                         auto dbResult = m_database->getMemberById(member->id);
00270                         OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00271                         auto memberArray =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00272                         oatpp::Object<MemberDto> currentMember;
00273
00274                         if(memberArray->size() == 1)
00275                             currentMember = memberArray[0];
00276                         else if(memberArray->size() > 1)
```

```
00277                           {
00278                               auto status = primus::dto::StatusDto::createShared();
00279                               status->code = 500;
00280                               status->message = "More than one user with same id";
00281                               status->status = "CRICTICAL DATABASE ERROR";
00282                               return createDtoResponse(Status::CODE_500, status);
00283                           }
00284                           else
00285                           {
00286                               auto status = primus::dto::StatusDto::createShared();
00287                               status->code = 404;
00288                               status->message = "Member was not found";
00289                               status->status = "NOT FOUND";
00290                               return createDtoResponse(Status::CODE_404, status);
00291                           }
00292
00293
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Old member
data:");
00294
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - ID: %d",
currentMember->id.operator v_uint32());
00295
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - First Name:
%s", currentMember->firstName->c_str());
00296
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Last Name:
%s", currentMember->lastName->c_str());
00297
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Email: %s",
currentMember->email->c_str());
00298
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Phone
Number: %s", currentMember->phoneNumber->c_str());
00299
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Birth Date:
%s", currentMember->birthDate->c_str());
00300
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Create
Date: %s", currentMember->createDate->c_str());
00301
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Notes: %s",
currentMember->notes->c_str());
00302
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Active:
%s", currentMember->active ? "true" : "false");
00303
00304                           member->firstName = member->firstName == nullptr ? "" :
member->firstName;
00305                           member->lastName = member->lastName == nullptr ? "" :
member->lastName;
00306                           member->email = member->email == nullptr ? "" :
member->email;
00307                           member->phoneNumber = member->phoneNumber == nullptr ? ""
: member->phoneNumber;
00308                           member->birthDate = member->birthDate == nullptr ? "" :
member->birthDate;
00309                           member->createDate = member->createDate == nullptr ? "" :
member->createDate;
00310                           member->notes = member->notes == nullptr ? "" :
member->notes;
00311                           member->active = member->active == nullptr ? "" :
member->active;
00312
00313
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "New member
data:");
00314
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - ID: %d",
member->id.operator v_uint32());
00315
OATPP LOGI(primus::constants::apicontroller::member endpoint::logName, "  - First Name:
%s", member->firstName->c_str());
00316
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Last Name:
%s", member->lastName->c_str());
```

```
00317
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Email: %s",
member->email->c_str());
00318
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Phone
Number: %s", member->phoneNumber->c_str());
00319
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Birth Date:
%s", member->birthDate->c_str());
00320
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Create
Date: %s", member->createDate->c_str());
00321
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Notes: %s",
member->notes->c_str());
00322
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "  - Active:
%s", member->active ? "true" : "false");
00323                     }
00324
00325                     auto dbResult = m_database->updateMember(member);
00326                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00327
00328
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Updated member
with id: %d", member->id.operator v_uint32());
00329
00330                     return getMemberById(member->id);
00331                 }
00332
00333                 ENDPOINT("GET", "/api/members/count/{attribute}", getMemberCount,
PATH(oatpp::String, attribute))
00334                 {
00335
00336
00337                     std::shared_ptr<oatpp::orm::QueryResult> dbResult;
00338
00339                     if (attribute == oatpp::String("all"))
00340                     {
00341                         dbResult = m_database->getMemberCountAll();
00342                     }
00343                     else if (attribute == oatpp::String("active"))
00344                     {
00345                         dbResult = m_database->getMemberCountActive();
00346                     }
00347                     else if (attribute == oatpp::String("inactive"))
00348                     {
00349                         dbResult = m_database->getMemberCountInactive();
00350                     }
00351                     else
00352                     {
00353
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received count
request for members with attribute %s, which is not an available attribute",
attribute->c_str());
00354
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Returning CODE
500: Bad Request. Available options: all, active, inactive");
00355
00356                         auto status = primus::dto::StatusDto::createShared();
00357                         status->code = 404;
00358                         status->message = "Received request for a count of members
with invalid attribute. Available options: all, active, inactive";
00359                         status->status = "INVALID ATTRIBUTE";
00360                         return createDtoResponse(Status::CODE_404, status);
00361                     }
00362
00363
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get count of %s members", attribute->c_str());
00364                     OATPP ASSERT HTTP(dbResult->isSuccess(), Status::CODE 500,
dbResult->getErrorMessage());
00365
00366                     auto count =
dbResult->fetch<oatpp::Vector<oatpp::Object<UInt32Dto>>>();
00367
```

```
00368
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Processed
request to get count of %s members. Total count: %d", attribute->c_str(),
count[0]->value.operator v_uint32());
00369
00370                     return createDtoResponse(Status::CODE_200, count[0]);
00371                 }
00372
00373                 ENDPOINT("GET", "/api/member/{memberId}/count/{attribute}",
getCountOfAttributeForMember, PATH(oatpp::UInt32, memberId), PATH(oatpp::String,
attribute))
00374                 {
00375                     std::shared_ptr<oatpp::orm::QueryResult> dbResult;
00376
00377                     if (attribute == oatpp::String("attendances"))
00378                     {
00379                         dbResult =
m_database->getCountOfAttendancesOfMember(memberId);
00380                     }
00381                     else
00382                     {
00383
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received count
request for count with attribute %s, which is not an available attribute",
attribute->c_str());
00384
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Returning CODE
500: Bad Request. Available options: all, active, inactive");
00385
00386                         auto status = primus::dto::StatusDto::createShared();
00387                         status->code = 404;
00388                         status->message = "Received request for a count with invalid
attribute. ";
00389                         status->status = "INVALID ATTRIBUTE";
00390                         return createDtoResponse(Status::CODE_404, status);
00391                     }
00392
00393                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00394
00395                     auto count =
dbResult->fetch<oatpp::Vector<oatpp::Object<UInt32Dto>>>();
00396
00397                     return createDtoResponse(Status::CODE_200, count[0]);
00398                 }
00399
00400                 ENDPOINT("POST",
"/api/member/{memberId}/department/add/{departmentId}",
createMemberDepartmentAssociation, PATH(oatpp::UInt32, memberId), PATH(oatpp::UInt32,
departmentId))
00401                 {
00402
00403
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to add member with id %d to department with id %d", memberId.operator v uint32(),
departmentId.operator v_uint32());
00404
00405                     std::shared_ptr<OutgoingResponse>           ret;
00406                     oatpp::Vector<oatpp::Object<DepartmentDto>> departments;
00407                     oatpp::Vector<oatpp::Object<MemberDto>>     member;
00408                     std::shared ptr<oatpp::orm::QueryResult>    dbResult;
00409
00410
00411                     dbResult = m_database->getDepartmentById(departmentId);
00412                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00413                     departments =
dbResult->fetch<oatpp::Vector<oatpp::Object<DepartmentDto>>>();
00414                     OATPP_ASSERT_HTTP(departments->size() != 0, Status::CODE_404,
"Department not found");
00415                     OATPP_ASSERT_HTTP(!(departments->size() > 1),
Status::CODE 404, "Critical database error: More than 1 department with id %d",
departmentId.operator v_uint32());
00416
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Department
found: %d | %s", departments[0]->id.operator v_uint32(), departments[0]->name->c_str());
00417
```

```
00418
00419
00420
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Creating
member-department association");
00421                       dbResult =
m_database->associateDepartmentWithMember(departmentId, memberId);
00422                       auto foo = dbResult->getErrorMessage();
00423
00424                       if (!dbResult->isSuccess())
00425                       {
00426                           oatpp::Object<primus::dto::StatusDto> ret =
primus::dto::StatusDto::createShared();
00427
00428                           oatpp::String verboseMessage = "Database operation did not
succeed. This might be because the member is already associated with the department. The
database error message was the following: ";
00429                           verboseMessage->append(dbResult->getErrorMessage());
00430
00431                           ret->code = 500;
00432                           ret->message = verboseMessage;
00433                           ret->status = "Member likely already in department";
00434
00435                           return createDtoResponse(Status::CODE_500, ret);
00436                       }
00437
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName,
"member-department association successfully created");
00438
00439                       auto status = primus::dto::StatusDto::createShared();
00440                       status->code = 200;
00441                       status->message = "member-department association successfully
created";
00442                       status->status = "member-department association successfully
created";
00443                       return createDtoResponse(Status::CODE_200, status);
00444                   }
00445                   ENDPOINT("DELETE",
"/api/member/{memberId}/department/remove/{departmentId}",
deleteMemberDepartmentDisassociation, PATH(oatpp::UInt32, memberId),
PATH(oatpp::UInt32, departmentId))
00446                   {
00447
00448
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to remove member with id %d from department with id %d", memberId.operator
v_uint32(), departmentId.operator v_uint32());
00449
00450                       std::shared_ptr<OutgoingResponse>           ret;
00451                       oatpp::Vector<oatpp::Object<DepartmentDto>> departments;
00452                       oatpp::Vector<oatpp::Object<MemberDto>>     member;
00453                       std::shared_ptr<oatpp::orm::QueryResult>    dbResult;
00454
00455
00456                       dbResult = m database->getDepartmentById(departmentId);
00457                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00458                       departments =
dbResult->fetch<oatpp::Vector<oatpp::Object<DepartmentDto>>>();
00459                       OATPP_ASSERT_HTTP(departments->size() != 0, Status::CODE_404,
"Department not found");
00460                       OATPP_ASSERT_HTTP(!(departments->size() > 1),
Status::CODE_404, "Critical database error: More than 1 department with id %d",
departmentId.operator v_uint32());
00461
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Department
found: %d | %s", departments[0]->id.operator v uint32(), departments[0]->name->c str());
00462
00463                       auto memberStatus =
primus::assert::assertMemberExists(memberId);
00464                       if (memberStatus->code != 200)
00465                           return createDtoResponse(Status::CODE 500, memberStatus);
00466
00467
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Disassociating
member and department");
```

```
00468                       dbResult =
m_database->disassociateDepartmentFromMember(departmentId, memberId);
00469                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00470
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "member and
department successfully disassociated");
00471
00472                       memberStatus = primus::dto::StatusDto::createShared();
00473
00474                       memberStatus->code = 200;
00475                       memberStatus->message = "member and department successfully
disassociated";
00476                       memberStatus->status = "member and department successfully
disassociated";
00477
00478                       return createDtoResponse(Status::CODE_200, memberStatus);
00479                   }
00480
00481                   ENDPOINT("POST", "/api/member/{memberId}/address/add",
createMemberAddressAssociation, PATH(oatpp::UInt32, memberId),
BODY_DTO(oatpp::Object<AddressDto>, address))
00482                   {
00483
00484
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to set address for member with id %d", memberId.operator v_uint32());
00485
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Address
data:");
00486
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "- Street: %s",
address->street->c_str());
00487
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "- City: %s",
address->city->c_str());
00488
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "- Postal code:
%s", address->postalCode->c_str());
00489
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "- Country: %s",
address->country->c_str());
00490
00491                       auto status = primus::assert::assertMemberExists(memberId);
00492                       if (status->code != 200)
00493                       {
00494                           return createDtoResponse(Status::CODE_500, status);
00495                       }
00496
00497                       auto dbResult = m_database->createAddress(address);
00498
00499                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00500
00501                       oatpp::UInt32 addressId =
oatpp::sqlite::Utils::getLastInsertRowId(dbResult->getConnection());
00502
00503                       oatpp::Vector<oatpp::Object<AddressDto>> foundAddresses;
00504                       oatpp::Object<AddressDto> retAddress;
00505
00506                       if (addressId == 0)
00507                       {
00508
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Address already
exists. Proceeding to return existing id");
00509
00510                           dbResult =
m_database->findAddressByDetails(address->street, address->city, address->postalCode,
address->country);
00511                       }
00512                       else
00513                       {
00514
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Address created
with id %d", addressId.operator v_uint32());
00515
00516                           dbResult = m_database->getAddressById(addressId);
```

```
00517                     }
00518                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
"Unknown Error");
00519
00520                     foundAddresses =
dbResult->fetch<oatpp::Vector<oatpp::Object<AddressDto>>>();
00521                     retAddress = foundAddresses[0];
00522
00523
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Creating
member-address association");
00524                     dbResult =
m_database->associateAddressWithMember(retAddress->id, memberId);
00525                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
"Unknown Error");
00526
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "member-address
association was successfully created");
00527
00528                     return createDtoResponse(Status::CODE_200, retAddress);
00529                 }
00530                 ENDPOINT("DELETE",
"/api/member/{memberId}/address/remove/{addressId}",
deleteMemberAddressDisassociation, PATH(oatpp::UInt32, memberId), PATH(oatpp::UInt32,
addressId))
00531                 {
00532
00533
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to remove member with id %d from address with id %d", memberId.operator v_uint32(),
addressId.operator v_uint32());
00534
00535                     std::shared_ptr<OutgoingResponse>           ret;
00536                     oatpp::Vector<oatpp::Object<AddressDto>>    addresses;
00537                     oatpp::Vector<oatpp::Object<MemberDto>>     member;
00538                     std::shared_ptr<oatpp::orm::QueryResult>    dbResult;
00539
00540
00541                     dbResult = m_database->getAddressById(addressId);
00542                     OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00543                     addresses =
dbResult->fetch<oatpp::Vector<oatpp::Object<AddressDto>>>();
00544                     OATPP_ASSERT_HTTP(addresses->size() != 0, Status::CODE_404,
"address not found");
00545                     OATPP_ASSERT_HTTP(!(addresses->size() > 1), Status::CODE_404,
"Critical database error: More than 1 address with id %d", departmentId.operator
v_uint32());
00546
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Address
found");
00547
00548                     {
00549                         auto status =
primus::assert::assertMemberExists(memberId);
00550                         if (status->code != 200)
00551                         {
00552                             return createDtoResponse(Status::CODE_500, status);
00553                         }
00554                     }
00555
00556
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Disassociating
member and address");
00557                     dbResult =
m_database->disassociateAddressFromMember(addressId, memberId);
00558                     OATPP ASSERT HTTP(dbResult->isSuccess(), Status::CODE 500,
dbResult->getErrorMessage());
00559
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "member and
department successfully disassociated");
00560
00561
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Checking for
other members using the address...");
00562                     dbResult = m_database->getMembersByAddress(addressId);
```

```
00563                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00564
00565                       member =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00566
00567
00568                       if (member->size() == 0)
00569                       {
00570
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "No other member
using the address.");
00571
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Proceeding with
deletion of address");
00572
00573                           dbResult = m_database->deleteAddress(addressId);
00574                           OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00575
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Address has
been deleted");
00576                       }
00577                       else
00578                       {
00579
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Other members
are associated with address");
00580
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "keeping address
in database");
00581                       }
00582                       auto status = primus::dto::StatusDto::createShared();
00583                       status->code = 200;
00584                       status->message = "member and department successfully
disassociated";
00585                       status->status = "member and department successfully
disassociated";
00586                       return createDtoResponse(Status::CODE_200, status);
00587                   }
00588
00589                   ENDPOINT("POST",
"/api/member/{memberId}/attendance/{dateOfAttendance}", addMemberAttendance,
PATH(oatpp::UInt32, memberId), PATH(oatpp::String, dateOfAttendance))
00590                   {
00591
00592
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request set member attendance for member with id %d", memberId.operator v_uint32());
00593
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Date of
attendance: %s", dateOfAttendance->c_str());
00594
00595                       {
00596                           auto status =
primus::assert::assertMemberExists(memberId);
00597
00598                           if (status->code != 200)
00599                               return createDtoResponse(Status::CODE_500, status);
00600                       }
00601
00602
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member found");
00603
00604                       auto dbResult = m_database->createMemberAttendance(memberId,
dateOfAttendance);
00605                       if (!dbResult->isSuccess())
00606                       {
00607                           auto dbResult =
m_database->hasMemberAttendedOnDate(memberId, dateOfAttendance);
00608                           OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE 500, dbResult->getErrorMessage());
00609                           auto foo = dbResult->getErrorMessage();
00610                           auto attendanceOnDate =
dbResult->fetch<oatpp::Vector<oatpp::Object<UInt32Dto>>>();
00611
00612                           if(attendanceOnDate[0]->value > 0)
```

```
00613                           {
00614                               auto status = primus::dto::StatusDto::createShared();
00615                               status->code = 403;
00616                               status->message = "Attendance Already set";
00617                               status->status = "Already set";
00618                               return createDtoResponse(Status::CODE_200, status);
00619                           }
00620                       }
00621
00622                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00623
00624
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member
attendance was set for date %s", dateOfAttendance->c_str());
00625
00626                       auto status = primus::dto::StatusDto::createShared();
00627                       status->code = 200;
00628                       status->message = "Attendance set";
00629                       status->status = "Attendance set";
00630                       return createDtoResponse(Status::CODE_200, status);
00631                   }
00632                   ENDPOINT("DELETE",
"/api/member/{memberId}/attendance/{dateOfAttendance}", deleteMemberAttendance,
PATH(oatpp::UInt32, memberId), PATH(oatpp::String, dateOfAttendance))
00633                   {
00634
00635
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request remove member attendance for member with id %d", memberId.operator v_uint32());
00636
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Date of
attendance: %s", dateOfAttendance->c_str());
00637
00638                       std::shared_ptr<oatpp::orm::QueryResult> dbResult =
m_database->getMemberById(memberId); // Wheather or not the member exists
00639                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00640
00641                       auto members =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00642                       OATPP_ASSERT_HTTP(members->size() > 0, Status::CODE_404,
"Member not found");
00643                       OATPP_ASSERT_HTTP(members->size() < 2, Status::CODE_500,
"Critical database error: more than one member with given id");
00644
00645
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member found");
00646
00647                       dbResult = m_database->deleteMemberAttendance(memberId,
dateOfAttendance);
00648                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00649
00650
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member
attendance was removed for date %s", dateOfAttendance->c_str());
00651
00652                       auto status = primus::dto::StatusDto::createShared();
00653                       status->code = 200;
00654                       status->message = "Attendance removed";
00655                       status->status = "Attendance removed";
00656                       return createDtoResponse(Status::CODE_200, status);
00657                   }
00658
00659                   ENDPOINT("GET", "/api/member/{memberId}/fee", getMemberFee,
00660                       PATH(oatpp::UInt32, memberId))
00661                   {
00662
00663
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to calculate the member fee for member with id %d.", memberId.operator v_uint32());
00664
00665                       std::shared_ptr<oatpp::orm::QueryResult> dbResult;
00666                       oatpp::Vector<oatpp::Object<MemberDto>> members;
00667                       oatpp::Vector<oatpp::Object<DepartmentDto>> departments;
00668                       auto memberFee = UInt32Dto::createShared();
```

```
00669
00670                       dbResult = m_database->getMemberById(memberId);
00671                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00672
00673                       members =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00674                       OATPP_ASSERT_HTTP(members->size() > 0, Status::CODE_404,
"Member not found");
00675                       OATPP_ASSERT_HTTP(members->size() < 2, Status::CODE_500,
"Critical database error: more than one member with given id");
00676
00677
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member was
found.", memberId.operator v_uint32());
00678
00679                       dbResult = m_database->getMemberDepartments(memberId,
oatpp::UInt32(3), oatpp::UInt32(static_cast<unsigned int>(0)));
00680                       OATPP_ASSERT_HTTP(dbResult->isSuccess(), Status::CODE_500,
dbResult->getErrorMessage());
00681                       departments =
dbResult->fetch<oatpp::Vector<oatpp::Object<DepartmentDto>>>();
00682
00683                       if (departments->size() < 1) // No department
00684                           memberFee->value =
primus::constants::pricing::DepartmentPrices::None;
00685                       else if (departments->size() > 1) // multiple departments
00686                           memberFee->value =
primus::constants::pricing::DepartmentPrices::Multiple;
00687                       else if (departments->size() == 1) // Only one department
00688                           switch (departments[0]->id)
00689                           {
00690                           case 1: // Bogenschiessen
00691                               memberFee->value =
primus::constants::pricing::DepartmentPrices::Bogenschiessen;
00692                               break;
00693                           case 2: // Luftdruck
00694                               memberFee->value =
primus::constants::pricing::DepartmentPrices::Luftdruck;
00695                               break;
00696                           case 3: // Schusswaffen
00697                               memberFee->value =
primus::constants::pricing::DepartmentPrices::Schusswaffen;
00698                               break;
00699                           }
00700                       else
00701                       {
00702                           auto status = primus::dto::StatusDto::createShared();
00703
00704                           status->status = "UNKNOWN ERROR";
00705                           status->code = 500;
00706                           status->message = "Failed while trying to determine
membership fee";
00707                           return createDtoResponse(Status::CODE_500, status);
00708                       }
00709
00710
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member is in %d
departments.", departments->size());
00711
OATPP LOGI(primus::constants::apicontroller::member endpoint::logName, "Fee is %d euro",
memberFee->value.operator v_uint32());
00712
00713
00714
00715                       return createDtoResponse(Status::CODE_200, memberFee);
00716                   }
00717
00718                   ENDPOINT("GET", "/api/member/{memberId}/list/{attribute}",
getMemberList,
00719                       PATH(oatpp::UInt32, memberId), PATH(oatpp::String,
attribute), QUERY(oatpp::UInt32, limit), QUERY(oatpp::UInt32, offset))
00720                   {
00721
00722
00723                       std::shared_ptr<oatpp::orm::QueryResult> dbResult;
00724                       std::shared_ptr<OutgoingResponse> ret;
```

```
00725
00726                     {
00727                         std::shared_ptr<oatpp::orm::QueryResult> dbResult =
m_database->getMemberById(memberId);
00728                         OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00729                         oatpp::Vector<oatpp::Object<MemberDto>> members =
dbResult->fetch<oatpp::Vector<oatpp::Object<MemberDto>>>();
00730                         OATPP_ASSERT_HTTP(members->size() > 0, Status::CODE_404,
"Member not found");
00731                         OATPP_ASSERT_HTTP(members->size() < 2, Status::CODE_500,
"Critical database error: more than one member with given id");
00732                     }
00733
00734                     if (attribute == oatpp::String("addresses"))
00735                     {
00736
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list addresses associated with member id %d. Limit: %d, Offset: %d",
memberId.operator v_uint32(), limit.operator v_uint32(), offset.operator v_uint32());
00737
00738                         dbResult = m_database->getMemberAddresses(memberId,
limit, offset);
00739                         OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00740
00741                         auto items =
dbResult->fetch<oatpp::Vector<oatpp::Object<AddressDto>>>();
00742
00743                         auto page =
PageDto<oatpp::Object<AddressDto>>::createShared();
00744
00745                         page->offset = offset;
00746                         page->limit = limit;
00747                         page->count = items->size();
00748                         page->items = items;
00749
00750
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Processed
request to get a list of members with %s. Limit: %d, Offset: %d. Returned %d items",
attribute->c_str(), limit.operator v_uint32(), offset.operator v_uint32(),
page->count.operator v_uint32());
00751
00752                         ret = createDtoResponse(Status::CODE_200, page);
00753                     }
00754                     else if (attribute == oatpp::String("departments"))
00755                     {
00756
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list departments associated with member id %d. Limit: %d, Offset: %d",
memberId.operator v_uint32(), limit.operator v_uint32(), offset.operator v_uint32());
00757
00758                         dbResult = m_database->getMemberDepartments(memberId,
limit, offset);
00759                         OATPP ASSERT HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00760
00761                         auto items =
dbResult->fetch<oatpp::Vector<oatpp::Object<DepartmentDto>>>();
00762
00763
00764                         auto page =
PageDto<oatpp::Object<DepartmentDto>>::createShared();
00765
00766                         page->offset = offset;
00767                         page->limit = limit;
00768                         page->count = items->size();
00769                         page->items = items;
00770
00771                         ret = createDtoResponse(Status::CODE_200, page);
00772                     }
00773                     else if (attribute == oatpp::String("attendances"))
00774                     {
00775
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list attendances associated with member id %d. Limit: %d, Offset: %d",
memberId.operator v_uint32(), limit.operator v_uint32(), offset.operator v_uint32());
```

```
00776
00777                           dbResult = m_database->getAttendancesOfMember(memberId,
limit, offset);
00778                           OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00779
00780                           auto items =
dbResult->fetch<oatpp::Vector<oatpp::Object<DateDto>>>();
00781
00782                           auto page =
PageDto<oatpp::Object<DateDto>>::createShared();
00783
00784                           page->offset = offset;
00785                           page->limit = limit;
00786                           page->count = items->size();
00787                           page->items = items;
00788
00789                           ret = createDtoResponse(Status::CODE_200, page);
00790                       }
00791                       else
00792                       {
00793
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to get a list of members with %s, which is not an available attribute",
attribute->c_str());
00794
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Returning CODE
500: Bad Request. Available options: addresses");
00795
00796                           auto status = primus::dto::StatusDto::createShared();
00797
00798                           oatpp::String verboseMessage = "Received request to get a
list of members with invalid attribute: ";
00799                           verboseMessage->append(attribute);
00800
00801
00802                           status->code = 404;
00803                           status->message = verboseMessage;
00804                           status->status = "INVALID ATTRIBUTE";
00805                           return createDtoResponse(Status::CODE_404, status);
00806                       }
00807
00808
00809
00810                       return ret;
00811                   }
00812
00813                   ENDPOINT("GET", "api/member/{memberId}/weaponpurchase/",
canMemberBuyWeapon,
00814                       PATH(oatpp::UInt32, memberId))
00815                   {
00816
00817
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Received
request to check if member with id %d", memberId.operator v uint32());
00818
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "is allowed to
purchase a weapon");
00819                       {
00820                           auto status =
primus::assert::assertMemberExists(memberId);
00821
00822                           if (status->code != 200)
00823                               return createDtoResponse(Status::CODE_500, status);
00824                       }
00825
OATPP_LOGI(primus::constants::apicontroller::member endpoint::logName, "Member was
found");
00826
00827                       {
00828                           std::shared_ptr<oatpp::orm::QueryResult> dbResult;
00829                           oatpp::Vector<oatpp::Object<UInt32Dto>> count;
00830                           auto ret = BooleanDto::createShared();
00831
00832                           dbResult =
m_database->getCountOfMemberAttendancesInLastYear(memberId);
```

```
00833                           OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00834                           count =
dbResult->fetch<oatpp::Vector<oatpp::Object<UInt32Dto>>>();
00835
00836
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Checking first
condition of weapon purchase...");
00837
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member attended
%d sessions last year.", count[0]->value.operator v_uint32());
00838
00839                           if (count[0]->value >= 18)
00840                           {
00841                               ret->value = true;
00842
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Which allowes
him to purchase a weapon");
00843
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Returning
true");
00844                           }
00845                           else
00846                           {
00847
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member does not
have the yearly attendance to purchase a weapon");
00848
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Checking for
secondary-condition (monthly attendance x1)");
00849
00850                               dbResult =
m_database->countDistinctAttendentMontsWithinLastYear(memberId);
00851                               OATPP_ASSERT_HTTP(dbResult->isSuccess(),
Status::CODE_500, dbResult->getErrorMessage());
00852                               count =
dbResult->fetch<oatpp::Vector<oatpp::Object<UInt32Dto>>>();
00853
00854                               if (count[0]->value == 12)
00855                               {
00856
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member attended
at least one session per month");
00857
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Returning
true");
00858                                   ret->value = true;
00859                               }
00860                               else
00861                               {
00862
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Member attended
less than one session per month");
00863
OATPP_LOGI(primus::constants::apicontroller::member_endpoint::logName, "Returning
false");
00864                                   ret->value = false;
00865                               }
00866                           }
00867
00868                           return createDtoResponse(Status::CODE_200, ret);
00869                       }
00870                   }
00871                   // Endpoint Infos
00872
00873                   ENDPOINT_INFO(getMembersList)
00874                   {
00875                       info->name = "getMembersList";
00876                       info->summary = "Get a list of members based on attribute";
00877                       info->description = "This endpoint retrieves a list of members
based on the provided attribute. Available attributes are: all, active, inactive,
birthday.";
00878                       info->path = "/api/members/list/{attribute}";
00879                       info->method = "GET";
00880                       info->addTag("Members");
00881                       info->addTag("List");
```

```
00882                       info->pathParams["attribute"].description = "Attribute to
filter members (options: all, active, inactive, birthday)";
00883                       info->queryParams["limit"].description = "Maximum number of
items to return";
00884                       info->queryParams["offset"].description = "Number of items to
skip before starting to collect the response items";
00885
info->addResponse<oatpp::Object<PageDto<oatpp::Vector<oatpp::Object<MemberDto>>>>>(Sta
tus::CODE_200, "application/json");
00886                       info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
00887                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00888                   }
00889
00890                   ENDPOINT_INFO(activateMember) {
00891                       info->name = "activateMember";
00892                       info->summary = "Activate a member by ID";
00893                       info->description = "This endpoint activates a member with the
provided ID.";
00894                       info->path = "/api/member/{id}/activate";
00895                       info->method = "UPDATE";
00896                       info->addTag("Member");
00897                       info->pathParams["id"].description = "Identifier of the member
to activate";
00898                       info->addResponse<Object<StatusDto>>(Status::CODE_200,
"application/json");
00899                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00900                   }
00901
00902                   ENDPOINT_INFO(deactivateMember) {
00903                       info->name = "deactivateMember";
00904                       info->summary = "Deactivate a member by ID";
00905                       info->description = "This endpoint deactivates a member with the
provided ID.";
00906                       info->path = "/api/member/{id}/deactivate";
00907                       info->method = "UPDATE";
00908                       info->addTag("Member");
00909                       info->pathParams["id"].description = "Identifier of the member
to deactivate";
00910                       info->addResponse<Object<StatusDto>>(Status::CODE_200,
"application/json");
00911                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00912                   }
00913
00914                   ENDPOINT_INFO(getMemberById) {
00915                       info->name = "getMemberById";
00916                       info->summary = "Get a member by ID";
00917                       info->description = "This endpoint retrieves a member with the
provided ID.";
00918                       info->path = "/api/member/{id}";
00919                       info->method = "GET";
00920                       info->addTag("Member");
00921                       info->pathParams["id"].description = "Identifier of the member
to retrieve";
00922
info->addResponse<oatpp::Vector<oatpp::Object<MemberDto>>>(Status::CODE_200,
"application/json");
00923                       info->addResponse<Object<StatusDto>>(Status::CODE 404,
"application/json");
00924                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00925                   }
00926
00927                   ENDPOINT INFO(createMember)
00928                   {
00929                       info->name = "createMember";
00930                       info->summary = "Create a new member";
00931                       info->description = "This endpoint creates a new member with the
provided data.";
00932                       info->path = "/api/member";
00933                       info->method = "POST";
00934                       info->addTag("Member");
00935                       info->bodyContentType = "application/json";
```

```
00936
info->addResponse<oatpp::Object<MemberDto>>(Status::CODE_200, "application/json");
00937                     info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00938                 }
00939
00940                 ENDPOINT_INFO(updateMember)
00941                 {
00942                     info->name = "updateMember";
00943                     info->summary = "Update an existing member";
00944                     info->description = "This endpoint updates an existing member
with the provided data.";
00945                     info->path = "/api/member";
00946                     info->method = "PUT";
00947                     info->addTag("Member");
00948                     info->bodyContentType = "application/json";
00949
info->addResponse<oatpp::Object<MemberDto>>(Status::CODE_200, "application/json");
00950                     info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00951                 }
00952
00953                 ENDPOINT_INFO(getMemberCount)
00954                 {
00955                     info->name = "getMemberCount";
00956                     info->summary = "Get the count of members based on attribute";
00957                     info->description = "This endpoint retrieves the count of
members based on the provided attribute. Available attributes are: all, active, inactive.";
00958                     info->path = "/api/members/count/{attribute}";
00959                     info->method = "GET";
00960                     info->addTag("Members");
00961                     info->addTag("Counts");
00962                     info->queryParams["attribute"].description = "Attribute to
filter members (options: all, active, inactive)";
00963                     info->addResponse<Object<UInt32Dto>>(Status::CODE_200,
"application/json");
00964                     info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
00965                     info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00966                 }
00967
00968                 ENDPOINT_INFO(createMemberDepartmentAssociation)
00969                 {
00970                     info->name = "createMemberDepartmentAssociation";
00971                     info->summary = "Create association between member and
department";
00972                     info->description = "This endpoint creates an association
between a member and a department.";
00973                     info->path =
"/api/member/{memberId}/department/add/{departmentId}";
00974                     info->method = "POST";
00975                     info->addTag("Member");
00976                     info->addTag("Department");
00977                     info->pathParams["memberId"].description = "ID of the member";
00978                     info->pathParams["departmentId"].description = "ID of the
department";
00979                     info->addResponse<Object<StatusDto>>(Status::CODE_200,
"application/json");
00980                     info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
00981                     info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00982                 }
00983
00984                 ENDPOINT_INFO(deleteMemberDepartmentDisassociation)
00985                 {
00986                     info->name = "deleteMemberDepartmentDisassociation";
00987                     info->summary = "Remove association between member and
department";
00988                     info->description = "This endpoint removes the association
between a member and a department.";
00989                     info->path =
"/api/member/{memberId}/department/remove/{departmentId}";
00990                     info->method = "DELETE";
00991                     info->addTag("Member");
00992                     info->addTag("Department");
```

```
00993                       info->pathParams["memberId"].description = "ID of the member";
00994                       info->pathParams["departmentId"].description = "ID of the
department";
00995                       info->addResponse<Object<StatusDto>>(Status::CODE_200,
"application/json");
00996                       info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
00997                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
00998                   }
00999
01000                   ENDPOINT_INFO(createMemberAddressAssociation)
01001                   {
01002                       info->name = "createMemberAddressAssociation";
01003                       info->summary = "Create association between member and
address";
01004                       info->description = "This endpoint creates an association
between a member and an address.";
01005                       info->path = "/api/member/{memberId}/address/add";
01006                       info->method = "POST";
01007                       info->addTag("Member");
01008                       info->addTag("Address");
01009                       info->pathParams["memberId"].description = "ID of the member";
01010                       info->addResponse<Object<AddressDto>>(Status::CODE_200,
"application/json");
01011                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
01012                   }
01013
01014                   ENDPOINT_INFO(deleteMemberAddressDisassociation)
01015                   {
01016                       info->name = "deleteMemberAddressDisassociation";
01017                       info->summary = "Remove association between member and
address";
01018                       info->description = "This endpoint removes the association
between a member and an address.";
01019                       info->path =
"/api/member/{memberId}/address/remove/{addressId}";
01020                       info->method = "DELETE";
01021                       info->addTag("Member");
01022                       info->addTag("Address");
01023                       info->pathParams["memberId"].description = "ID of the member";
01024                       info->pathParams["addressId"].description = "ID of the
address";
01025                       info->addResponse<Object<StatusDto>>(Status::CODE_200,
"application/json");
01026                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
01027                   }
01028
01029                   ENDPOINT_INFO(addMemberAttendance)
01030                   {
01031                       info->name = "addMemberAttendance";
01032                       info->summary = "Add attendance for a member on a specific date";
01033                       info->description = "This endpoint adds attendance for a member
on a specific date.";
01034                       info->path =
"/api/member/{memberId}/attendance/{dateOfAttendance}";
01035                       info->method = "POST";
01036                       info->addTag("Member");
01037                       info->addTag("Attendance");
01038                       info->pathParams["memberId"].description = "ID of the member";
01039                       info->pathParams["dateOfAttendance"].description = "Date of
the attendance (format: YYYY-MM-DD)";
01040                       info->addResponse<Object<StatusDto>>(Status::CODE_200,
"application/json");
01041                       info->addResponse<Object<StatusDto>>(Status::CODE 404,
"application/json");
01042                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
01043                   }
01044
01045                   ENDPOINT_INFO(deleteMemberAttendance)
01046                   {
01047                       info->name = "deleteMemberAttendance";
01048                       info->summary = "Remove attendance for a member on a specific
date";
```

```
01049                       info->description = "This endpoint removes attendance for a
member on a specific date.";
01050                       info->path =
"/api/member/{memberId}/attendance/{dateOfAttendance}";
01051                       info->method = "DELETE";
01052                       info->addTag("Member");
01053                       info->addTag("Attendance");
01054                       info->pathParams["memberId"].description = "ID of the member";
01055                       info->pathParams["dateOfAttendance"].description = "Date of
the attendance (format: YYYY-MM-DD)";
01056                       info->addResponse<Object<StatusDto>>(Status::CODE_200,
"application/json");
01057                       info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
01058                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
01059                   }
01060
01061                   ENDPOINT_INFO(getMemberFee)
01062                   {
01063                       info->name = "getMemberFee";
01064                       info->summary = "Calculate the member fee for a member";
01065                       info->description = "This endpoint calculates the membership
fee for a member based on their department.";
01066                       info->path = "/api/member/{memberId}/fee";
01067                       info->method = "GET";
01068                       info->addTag("Member");
01069                       info->addTag("Fee");
01070                       info->pathParams["memberId"].description = "ID of the member";
01071                       info->addResponse<Object<UInt32Dto>>(Status::CODE_200,
"application/json");
01072                       info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
01073                       info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
01074                   }
01075
01076                   ENDPOINT_INFO(getMemberList)
01077                   {
01078                       info->name = "getMemberList";
01079                       info->summary = "Get a list of information associated with a
member";
01080                       info->description = "This endpoint retrieves a list of
information associated with a member, such as addresses, departments, or attendances.";
01081                       info->path = "/api/member/{memberId}/list/{attribute}";
01082                       info->method = "GET";
01083                       info->addTag("Member");
01084                       info->addTag("List");
01085                       info->pathParams["memberId"].description = "ID of the member";
01086                       info->pathParams["attribute"].description = "Attribute to
retrieve (addresses, departments, attendances)";
01087                       info->queryParams["limit"].description = "Limit of items to
retrieve (default is 0)";
01088                       info->queryParams["offset"].description = "Offset for
pagination (default is 0)";
01089
info->addResponse<Object<PageDto<oatpp::Object<AddressDto>>>>(Status::CODE_200,
"application/json");
01090
info->addResponse<Object<PageDto<oatpp::Object<DepartmentDto>>>>(Status::CODE_200,
"application/json");
01091
info->addResponse<Object<PageDto<oatpp::Object<DateDto>>>>(Status::CODE_200,
"application/json");
01092                       info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
01093                       info->addResponse<Object<StatusDto>>(Status::CODE 500,
"application/json");
01094                   }
01095
01096                   ENDPOINT_INFO(canMemberBuyWeapon)
01097                   {
01098                       info->name = "canMemberBuyWeapon";
01099                       info->summary = "Check if a member is allowed to purchase a
weapon";
01100                       info->description = "This endpoint checks if a member is
eligible to purchase a weapon based on their attendance records.";
```

```
01101                     info->path = "/api/member/{memberId}/weaponpurchase/";
01102                     info->method = "GET";
01103                     info->addTag("Member");
01104                     info->addTag("Weapon");
01105                     info->pathParams["memberId"].description = "ID of the member";
01106                     info->addResponse<Object<BooleanDto>>(Status::CODE_200,
"application/json");
01107                     info->addResponse<Object<StatusDto>>(Status::CODE_404,
"application/json");
01108                     info->addResponse<Object<StatusDto>>(Status::CODE_500,
"application/json");
01109                 }
01110 
01111 
01112             };
01113 
01114 #include OATPP_CODEGEN_END(ApiController) // End API Controller codegen
01115 
01116         } // namespace member_endpoint
01117     } // apicontroller
01118 } // namespace namespace primus
01119 
01120 #endif // MEMBERCONTROLLER_HPP
```

StaticController.hpp

```
00001 #include "oatpp/web/server/api/ApiController.hpp"
00002 #include "oatpp/core/macro/codegen.hpp"
00003 #include "oatpp/core/macro/component.hpp"
00004 #include "oatpp/web/protocol/http/outgoing/BufferBody.hpp"
00005 #include <iostream>
00006 #include <fstream>
00007 #include <sstream>
00008 #include <chrono>
00009
00010 #include "general/constants.hpp"
00011
00012 namespace primus {
00013     namespace apicontroller {
00014         namespace static_endpoint {
00015
00016 #include OATPP_CODEGEN_BEGIN(ApiController)
00017             //  ____  _        _  _        ____                _                 _ _  ___
_ __
00018             // / ___|| |_ __ _| |_(_) ___ / ___|___  _ __ | |_ _ __ ___ | | | ___
00019             // \___ \| __/ _` | __| |/ __| |   / _ \| '_ \| __| '__/ _ \| | |/ _ \
'__|
00020             //  ___) | || (_| | |_| | (__| |__| (_) | | | | |_| | | (_) | | |  __/
|
00021             // |____/ \__\__,_|\__|_|\___|\____\___/|_| |_|\__|_|
\___/|_|_|\___|_|
00022             class StaticController : public oatpp::web::server::api::ApiController
00023             {
00024             public:
00025                 StaticController(OATPP_COMPONENT(std::shared_ptr<ObjectMapper>,
objectMapper))
00026                     : oatpp::web::server::api::ApiController(objectMapper)
00027                 {
00028
00029
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName,
"StaticController (oatpp::web::server::api::ApiController) initialized");
00030
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName,
"WEB_CONTENT_DIRECTORY: %s", WEB_CONTENT_DIRECTORY);
00031
00032                 }
00033
00034             public:
00035                 static std::shared_ptr<StaticController> createShared(
00036                     OATPP_COMPONENT(std::shared_ptr<ObjectMapper>, objectMapper)
00037                 )
00038                 {
00039                     return std::make_shared<StaticController>(objectMapper);
00040                 }
00041
00042                 ENDPOINT("GET", "/web/*", files,
00043                     REQUEST(std::shared_ptr<IncomingRequest>, request))
00044                 {
00045
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Received
request to serve file: %s", request->getPathTail()->c_str());
00046
00047                     // Ignore query parameters if present
00048                     auto pathTail = request->getPathTail();
00049                     auto queryPos = pathTail->find("?");
00050                     if (queryPos != std::string::npos) {
00051                         pathTail = pathTail->substr(0, queryPos);
00052                     }
00053
00054                     std::string filePath(WEB_CONTENT_DIRECTORY);
00055                     filePath.append("/");
00056
00057                     if (pathTail->empty()) {
00058                         filePath.append("index.html");
00059                     }
00060                     else {
00061                         filePath.append(*pathTail);
00062                     }
```

```
00063
00064
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Serving file:
%s", filePath.c_str());
00065
00066                     std::ifstream file(filePath, std::ios::binary);
00067                     if (file.good()) {
00068
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "File at %s
found", filePath.c_str());
00069
00070                         std::ostringstream content;
00071                         content << file.rdbuf();
00072                         file.close();
00073
00074
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Processed
request to serve file: %s", request->getPathTail()->c_str());
00075
00076                         return createResponse(Status::CODE_200, content.str());
00077                     }
00078                     else {
00079
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "File at %s was
not found", filePath.c_str());
00080
00081                         auto status = primus::dto::StatusDto::createShared();
00082
00083                         std::string verboseMessage = "File at \"";
00084                         verboseMessage.append(filePath.c_str());
00085                         verboseMessage.append("\" could not be found");
00086
00087                         status->code = 404;
00088                         status->message = verboseMessage;
00089                         status->status = "NOT FOUND";
00090                         return createDtoResponse(Status::CODE_404, status);
00091                     }
00092                 }
00093
00094
00095                 ENDPOINT("GET", "/", root,
00096                     REQUEST(std::shared_ptr<IncomingRequest>, request))
00097                 {
00098
00099
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Received
request for root");
00100
00101                     auto response = createResponse(Status::CODE_302, "Redirect");
00102                     response->putHeader("Location", "/web/");
00103
00104
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Redirecting to
/web/");
00105
00106
00107                     return response;
00108                 }
00109
00110                 ENDPOINT("GET", "/api/member/{memberId}/assets/profilepicture",
getAvatar, PATH(oatpp::String, memberId))
00111                 {
00112
00113
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Received
request to serve profile picture for member with id  %s", memberId->c_str());
00114
00115                     std::string filePath(USER_ASSETS);
00116                     std::string finalPath; filePath.append("/");
00117                     std::ostringstream content;
00118                     int choice;
00119
00120                     {
00121                         auto now = std::chrono::system_clock::now();
00122                         auto since_epoch = now.time_since_epoch();
00123                         auto millis =
std::chrono::duration_cast<std::chrono::milliseconds>(since_epoch).count();
```

```
00124                           choice = millis % 2;
00125                       }
00126
00127                       auto userStatus =
primus::assert::assertMemberExists(oatpp::utils::conversion::strToUInt32(memberId->c_s
tr()));
00128
00129                       if (userStatus->code != 200)
00130                       {
00131
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "User does not
exist. Returning default profile picture");
00132
00133                           filePath.append(choice == 1 ? "default-avatar-1.jpg" :
"default-avatar-2.jpg");
00134
00135                       }
00136                       else
00137                       {
00138
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "User found.
Looking for profile picture within directory");
00139                           filePath.append(memberId);
00140                           filePath.append(".jpg");
00141                       }
00142
00143                       finalPath = filePath;
00144
00145
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Serving file:
%s", filePath.c_str());
00146
00147                       std::ifstream file(filePath, std::ios::binary);
00148                       if (!file.good())
00149                       {
00150
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "File at %s was
not found", filePath.c_str());
00151
00152                           std::string filePath2(USER_ASSETS);
00153                           filePath2.append(choice == 1 ? "/default-avatar-1.jpg" :
"/default-avatar-2.jpg");
00154
00155                           finalPath = filePath2;
00156
00157                           std::ifstream file2(filePath2, std::ios::binary);
00158
00159
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "Proceeding to
serve file %s", filePath2.c_str());
00160
00161                           if (!file2.good())
00162                           {
00163
OATPP LOGE(primus::constants::apicontroller::static endpoint::logName, "Default profile
picture not found at %s", filePath2.c_str());
00164
00165                               auto status = primus::dto::StatusDto::createShared();
00166
00167                               oatpp::String verboseMessage = "No picture to serve.
Looked at given paths: ";
00168                               verboseMessage->append(filePath);
00169                               verboseMessage->append("   |   ");
00170                               verboseMessage->append(filePath2);
00171
00172                               status->code = 404;
00173                               status->message = verboseMessage;
00174                               status->status = "NOT FOUND";
00175                               return createDtoResponse(Status::CODE_404, status);
00176                           }
00177                           content << file2.rdbuf();
00178                       }
00179                       else
00180                       {
00181                           content << file.rdbuf();
00182                       }
00183
```

```
00184
OATPP_LOGI(primus::constants::apicontroller::static_endpoint::logName, "File at %s
found", finalPath.c_str());
00185
00186                     file.close();
00187
00188
00189
00190                     return createResponse(Status::CODE_200, content.str());
00191                 }
00192
00193                 // Endpoint Infos
00194
00195                 ENDPOINT_INFO(getAvatar) {
00196                     info->summary = "Get profile picture for a member";
00197                     info->path = "/api/member/{memberId}/assets/profilepicture";
00198                     info->pathParams.add("memberId",
oatpp::String::Class::getType());
00199                     info->addResponse<String>(Status::CODE_200, "image/jpeg");
00200
info->addResponse<Object<primus::dto::StatusDto>>(Status::CODE_404,
"application/json");
00201                     info->addTag("Member");
00202                     info->addTag("Static File");
00203                 }
00204
00205
00206                 ENDPOINT_INFO(files)
00207                 {
00208                     info->name = "files";
00209                     info->summary = "Serve static files";
00210                     info->description = "This endpoint serves static files from the
'/web' directory.";
00211                     info->path = "/web/*";
00212                     info->method = "GET";
00213                     info->addTag("Static File");
00214                     info->pathParams["*"].description = "File path relative to the
'/web' directory";
00215                     info->addResponse<String>(Status::CODE_200, "text/html");
00216
info->addResponse<Object<primus::dto::StatusDto>>(Status::CODE_404,
"application/json");
00217                 }
00218
00219                 ENDPOINT_INFO(root)
00220                 {
00221                     info->name = "root";
00222                     info->summary = "Redirect to web directory";
00223                     info->description = "This endpoint redirects requests to the
root URL to the '/web' directory.";
00224                     info->path = "/";
00225                     info->addTag("Static File");
00226                     info->method = "GET";
00227                     info->addResponse<String>(Status::CODE_302, "text/html");
00228                 }
00229             };
00230
00231 #include OATPP_CODEGEN_END(ApiController)
00232
00233         } // namespace static_endpoint
00234     } // namespace apicontroller
00235 } // namespace primus
```

DatabaseClient.hpp

```
00001 #ifndef DATABASE_CLIENT
00002 #define DATABASE_CLIENT
00003
00004 #include "oatpp-sqlite/orm.hpp"
00005 #include "oatpp/orm/SchemaMigration.hpp"
00006 #include "oatpp/orm/DbClient.hpp"
00007 #include "oatpp/core/macro/codegen.hpp"
00008
00009 #include "dto/DatabaseDtos.hpp"
00010 #include "general/constants.hpp"
00011
00012 namespace primus
00013 {
00014     namespace component
00015     {
00016 #include OATPP_CODEGEN_BEGIN(DbClient) //<- Begin Codegen
00017         //  ____        _        _                        ____ _ _                 _
00018         // |  _ \  __ _| |_ __ _| |__   __ _ ___  ___    / ___| (_) ___ _ __ | |_
00019         // | | | |/ _` | __/ _` | '_ \ / _` / __|/ _ \  | |   | | |/ _ \ '_ \| __|
00020         // | |_| | (_| | || (_| | |_) | (_| \__ \  __/  | |___| | |  __/ | | | |_
00021         // |____/ \__,_|\__\__,_|_.__/ \__,_|___/\___|  \____|_|_|\___|_| |_|\__|
00025         class DatabaseClient : public oatpp::orm::DbClient
00026         {
00027             typedef primus::dto::database::AddressDto       AddressDto;
00028             typedef primus::dto::database::DepartmentDto    DepartmentDto;
00029             typedef primus::dto::database::MemberDto        MemberDto;
00030         public:
00035             DatabaseClient(const std::shared_ptr<oatpp::orm::Executor>& executor)
00036                 : oatpp::orm::DbClient(executor)
00037             {
00038                 OATPP_LOGI(primus::constants::databaseclient::logName,
primus::constants::databaseclient::logSeperation);
00039                 OATPP_LOGI(primus::constants::databaseclient::logName,
"DatabaseClient(oatpp::orm::DbClient) initialized");
00040                 OATPP_LOGI(primus::constants::databaseclient::logName,
primus::constants::databaseclient::logSeperation);
00041
00042                 oatpp::orm::SchemaMigration migration(executor);
00043                 migration.addFile(1 /* start from version 1 */, DATABASE_MIGRATIONS
"/001_init.sql");
00044                 migration.migrate(); // <-- run migrations. This guy will throw on
error.
00045
00046                 auto version = executor->getSchemaVersion();
00047                 OATPP_LOGI(primus::constants::databaseclient::logName,"Migration
- OK. Version=%lld.", version);
00048             }
00049
00050             //                                  _
00051             //  _ __ ___   ___ _ __ ___ ___   | |__   ___ _ __
00052             // | '_ ` _ \ / _ \ '_ ` _ \ / _ \| '_ \ / _ \ '__|
00053             // | | | | | |  __/ | | | | | |_) |  __/ |
00054             // |_| |_| |_|\___|_| |_| |_|_.__/ \___|_|
00055
00062             QUERY(getMemberById, "SELECT * from Member WHERE id = :id;",
PARAM(oatpp::UInt32, id));
00063
00064
00071             QUERY(createMember,
00072                 "INSERT INTO Member (firstName, lastName, email, phoneNumber,
birthDate, createDate, notes, active) "
00073                 "SELECT :member.firstName, :member.lastName, :member.email,
:member.phoneNumber, :member.birthDate, DATE('now'), :member.notes, :member.active "
00074                 "WHERE NOT EXISTS (SELECT 1 FROM Member WHERE firstName =
:member.firstName AND lastName = :member.lastName AND email = :member.email AND birthDate
= :member.birthDate);",
00075                 PARAM(oatpp::Object<MemberDto>, member));
00076
00077
00078             QUERY(updateMember,
00079                 "UPDATE Member SET "
00080                 "firstName = :member.firstName, "
00081                 "lastName = :member.lastName, "
```

```
00082                 "email = :member.email, "
00083                 "phoneNumber = :member.phoneNumber, "
00084                 "birthDate = :member.birthDate, "
00085                 "notes = :member.notes, "
00086                 "active = :member.active "
00087                 "WHERE id = :member.id;",
00088                 PARAM(oatpp::Object<MemberDto>, member));
00089
00090             QUERY(findMemberIdByDetails,
00091                 "SELECT * FROM Member WHERE firstName = :firstName AND lastName =
:lastName AND email = :email AND birthDate = :birthDate;",
00092                 PARAM(oatpp::String, firstName),
00093                 PARAM(oatpp::String, lastName),
00094                 PARAM(oatpp::String, email),
00095                 PARAM(oatpp::String, birthDate));
00096
00097             QUERY(activateMember, "UPDATE Member SET active = 1 WHERE id = :id;",
PARAM(oatpp::UInt32, id));
00098
00099             QUERY(deactivateMember, "UPDATE Member SET active = 0 WHERE id = :id;",
PARAM(oatpp::UInt32, id));
00100
00101             //                                _
_
00102             //  _ __ ___   ___ _ __ ___ | |__   ___ _ __    ___ ___  _   _ _ __ |
|_ ___
00103             // | '_ ` _ \ / _ \ '_ ` _ \| '_ \ / _ \ '__|  / __/ _ \| | | | '_ \|
__/ __|
00104             // | | | | | |  __/ | | | | | |_) |  __/ |    | (_| (_) | |_| | | | |
|_\__ \
00105             // |_| |_| |_|\___|_| |_| |_|_.__/ \___|_|     \___\___/ \__,_|_|
|_|\__|___/
00106
00107             QUERY(getMemberCountAll, "SELECT COUNT(*) as value FROM Member");
00108
00109             QUERY(getMemberCountActive, "SELECT COUNT(*) as value FROM Member WHERE
active=1");
00110
00111             QUERY(getMemberCountInactive, "SELECT COUNT(*) as value FROM Member
WHERE active=0");
00112
00113             //                                          _                 _ _     _
00114             //  _ __ ___   ___ _ __ ___ | |__   ___ _ __  | (_)___| |_ ___
00115             // | '_ ` _ \ / _ \ '_ ` _ \| '_ \ / _ \ '__| | | / __| __/ __|
00116             // | | | | | |  __/ | | | | | |_) |  __/ |    | | \__ \ |_\__ \
00117             // |_| |_| |_|\___|_| |_| |_|_.__/ \___|_|    |_|_|___/\__|___/
00118
00119             QUERY(getMembersWithUpcomingBirthday,
00120                 " SELECT * from Member m "
00121                 " WHERE active = 1 AND strftime('%m-%d', m.birthDate) >=
strftime('%m-%d', 'now') "
00122                 " ORDER BY strftime('%m-%d', m.birthDate) ASC "
00123                 " LIMIT :limit OFFSET :offset; ",
00124                 PARAM(oatpp::UInt32, limit),
00125                 PARAM(oatpp::UInt32, offset));
00126
00127             QUERY(getAllMembers,
00128                 " SELECT * FROM Member "
00129                 " LIMIT :limit OFFSET :offset;",
00130                 PARAM(oatpp::UInt32, limit),
00131                 PARAM(oatpp::UInt32, offset));
00132
00133             QUERY(getActiveMembers,
00134                 " SELECT * FROM Member "
00135                 " WHERE active = 1 "
00136                 " ORDER BY id "
00137                 " LIMIT :limit OFFSET :offset;",
00138                 PARAM(oatpp::UInt32, limit),
00139                 PARAM(oatpp::UInt32, offset));
00140
00141             QUERY(getMembersByMostTraining,
00142                 " SELECT m.* "
00143                 " FROM Member m "
00144                 " JOIN( "
00145                 "     SELECT member_id, COUNT(*) AS attendance_count "
00146                 "     FROM Attendance"
00147                 "     WHERE date >= date('now', '-6 months') "
```

```
00148                 "     GROUP BY member_id "
00149                 "     ORDER BY attendance_count DESC "
00150                 "     LIMIT :limit OFFSET :offset "
00151                 " ) AS top_members ON m.id = top_members.member_id; ",
00152                 PARAM(oatpp::UInt32, limit),
00153                 PARAM(oatpp::UInt32, offset));
00154
00155             QUERY(getInactiveMembers,
00156                 " SELECT * FROM Member "
00157                 " WHERE active = 0 "
00158                 " ORDER BY id "
00159                 " LIMIT :limit OFFSET :offset;",
00160                 PARAM(oatpp::UInt32, limit),
00161                 PARAM(oatpp::UInt32, offset));
00162
00163             QUERY(getMembersByAddress, "SELECT Member.* FROM Member INNER JOIN
Address_Member ON Member.id = Address_Member.member_id WHERE Address_Member.address_id =
:addressId;", PARAM(oatpp::UInt32, addressId));
00164
00165             QUERY(getMembersByDepartment, "SELECT Member.* FROM Member INNER JOIN
Department_Member ON Member.id = Department_Member.member_id WHERE
Department_Member.department_id = :departmentId;", PARAM(oatpp::UInt32, departmentId));
00166
00167             //                                                  _                 _          __
00168             //  _ __ ___   ___ _ __ ___ ___ | |__   ___ _ __  (_)_ __   __ _  | |___ ___
00169             // | '_ ` _ \ / _ \ '_ ` _ \| '_ \ / _ \ '__| | | '_ \| |_ / _ \/ __|
00170             // | | | | | |  __/ | | | | | |_) |  __/ |    | | | | | |  _| (_) \__ \
00171             // |_| |_| |_|\___|_| |_| |_|_.__/ \___|_|    |_|_| |_|_|  \___/|___/
00172
00173             QUERY(getDepartmentsOfMember,
00174                 "SELECT d.* FROM Department d "
00175                 "JOIN MemberDepartmentRel mdr ON d.id = mdr.departmentID "
00176                 "WHERE mdr.memberID = :id;",
00177                 PARAM(oatpp::UInt32, id));
00178
00179             QUERY(getMemberAddresses,
00180                 " SELECT a.* FROM Address a "
00181                 " INNER JOIN Address_Member am ON a.id = am.address_id "
00182                 " WHERE am.member_id = :id "
00183                 " ORDER BY a.id "
00184                 " LIMIT :limit OFFSET :offset;",
00185                 PARAM(oatpp::UInt32, id),
00186                 PARAM(oatpp::UInt32, limit),
00187                 PARAM(oatpp::UInt32, offset));
00188
00189             QUERY(getMemberDepartments,
00190                 " SELECT d.* FROM Department d "
00191                 " INNER JOIN Department_Member dm ON d.id = dm.department_id "
00192                 " WHERE dm.member_id = :id "
00193                 " ORDER BY d.id ASC"
00194                 " LIMIT :limit OFFSET :offset;",
00195                 PARAM(oatpp::UInt32, id),
00196                 PARAM(oatpp::UInt32, limit),
00197                 PARAM(oatpp::UInt32, offset));
00198
00199             //       _                                  _                          _
00200             //    __| | ___ _ __   __ _ _ __| |_ _ __ ___ ___ _ __ | |_
00201             //   / _` |/ _ \ '_ \ / _` | '__| __| '_ ` _ \ / _ \ '_ \| __|
00202             //  | (_| |  __/ |_) | (_| | |  | |_| | | | | |  __/ | | | |_
00203             //   \__,_|\___| .__/ \__,_|_|   \__|_| |_| |_|\___|_| |_|\__|
00204             //             | |
00205
00206             QUERY(getDepartmentById,
00207                 " SELECT * FROM Department "
00208                 " WHERE id = :id; "
00209                 " ORDER BY id ASC",
00210                 PARAM(oatpp::UInt32, id));
00211
00212             //                _     _
00213             //    __ _  __| | __| |_ __ ___  ___ ___
00214             //   / _` |/ _` |/ _` | '__/ _ \/ __/ __|
00215             //  | (_| | (_| | (_| | | |  __/\__ \__ \
00216             //   \__,_|\__,_|\__,_|_|  \___||___/___/
00217
00218             QUERY(createAddress,
00219                 " INSERT INTO Address (postalCode, city, country, houseNumber,
street) "
```

```
00220                 " SELECT :address.postalCode, :address.city, :address.country,
:address.houseNumber, :address.street "
00221                 " WHERE NOT EXISTS (SELECT 1 FROM Address WHERE postalCode =
:address.postalCode AND city = :address.city AND country = :address.country AND houseNumber
= :address.houseNumber AND street = :address.street);",
00222                 PARAM(oatpp::Object<AddressDto>, address));
00223
00224             QUERY(getAddressById, "SELECT * FROM Address WHERE id = :id;",
PARAM(oatpp::UInt32, id));
00225
00226             QUERY(updateAddress,
00227                 "UPDATE Address SET "
00228                 "street = :address.street, "
00229                 "city = :address.city, "
00230                 "zipCode = :address.zipCode, "
00231                 "country = :address.country "
00232                 "WHERE id = :address.id;",
00233                 PARAM(oatpp::Object<AddressDto>, address));
00234
00235             QUERY(deleteAddress, "DELETE FROM Address WHERE id = :id AND id NOT IN
(SELECT address_id FROM Address_Member);", PARAM(oatpp::UInt32, id));
00236
00237             QUERY(findAddressByDetails,
00238                 "SELECT * FROM Address WHERE street = :street AND city = :city AND
postalCode = :postalCode AND country = :country;",
00239                 PARAM(oatpp::String, street),
00240                 PARAM(oatpp::String, city),
00241                 PARAM(oatpp::String, postalCode),
00242                 PARAM(oatpp::String, country));
00243
00244             //        _   _                      _
00245             //    ___| |_| |_ ___ _ __     __| | ___ _ __   ___ ___
00246             //   / _` | __| __/ _ \ '_ \ / _` |/ _` | '_ \ / __/ _ \
00247             // | (_| | |_| ||  __/ | | | (_| | (_| | | | | (_|  __/
00248             //  \__,_|\__|\__\___|_| |_|\__,_|\__,_|_| |_|\___\___|
00249
00250             QUERY(createMemberAttendance,
00251                 "INSERT INTO Attendance (member_id, date) "
00252                 "VALUES (:member_id, :date); ",
00253                 PARAM(oatpp::UInt32, member_id),
00254                 PARAM(oatpp::String, date));
00255
00256             QUERY(deleteMemberAttendance,
00257                 "DELETE FROM Attendance "
00258                 "WHERE member_id = :member_id AND date = :date;",
00259                 PARAM(oatpp::UInt32, member_id),
00260                 PARAM(oatpp::String, date));
00261
00262             QUERY(getAttendancesOfMember,
00263                 " SELECT date FROM Attendance "
00264                 " WHERE member_id = :member_id "
00265                 " ORDER BY date DESC "
00266                 " LIMIT :limit OFFSET :offset;",
00267                 PARAM(oatpp::UInt32, member_id),
00268                 PARAM(oatpp::UInt32, limit),
00269                 PARAM(oatpp::UInt32, offset));
00270
00271             QUERY(getCountOfAttendancesOfMember,
00272                 " SELECT COUNT(*) as value FROM Attendance "
00273                 " WHERE member_id = :member_id ",
00274                 PARAM(oatpp::UInt32, member_id));
00275
00276             QUERY(getMembersByAttendanceDate,
00277                 " SELECT member_id as value FROM Attendance "
00278                 " WHERE date = :date "
00279                 " ORDER BY date "
00280                 " LIMIT :limit OFFSET :offset;",
00281                 PARAM(oatpp::String, dateOfAttendance),
00282                 PARAM(oatpp::UInt32, limit),
00283                 PARAM(oatpp::UInt32, offset));
00284
00285             QUERY(hasMemberAttendedOnDate,
00286                 " SELECT COUNT(*) as value FROM Attendance "
00287                 " WHERE date = :date AND member_id = :member_id;",
00288                 PARAM(oatpp::UInt32, member_id),
00289                 PARAM(oatpp::String, date));
00290
```

```
            //     _                  _   _                   _        _     _
            //    (_)_   _ _ __   ___| |_(_) ___  _ __    | |_ __ _| |__ | | ___  ___
            //    | | | | | '_ \ / __| __| |/ _ \| '_ \   | __/ _` | '_ \| |/ _ \/ __|
            //    | | |_| | | | | (__| |_| | (_) | | | |  | || (_| | |_) | |  __/\__ \
            //   _/ |\__,_|_| |_|\___|\__|_|\___/|_| |_|  \__\__,_|_.__/|_|\___||___/
            // |__/

            QUERY(associateAddressWithMember, "INSERT INTO Address_Member (address_id, member_id) VALUES (:addressId, :memberId);", PARAM(oatpp::UInt32, addressId), PARAM(oatpp::UInt32, memberId));
            QUERY(disassociateAddressFromMember, "DELETE FROM Address_Member WHERE address_id = :addressId AND member_id = :memberId;", PARAM(oatpp::UInt32, addressId), PARAM(oatpp::UInt32, memberId));

            QUERY(associateDepartmentWithMember, "INSERT INTO Department_Member (department_id, member_id) VALUES (:departmentId, :memberId);", PARAM(oatpp::UInt32, departmentId), PARAM(oatpp::UInt32, memberId));
            QUERY(disassociateDepartmentFromMember, "DELETE FROM Department_Member WHERE department_id = :departmentId AND member_id = :memberId;", PARAM(oatpp::UInt32, departmentId), PARAM(oatpp::UInt32, memberId));

            // __        _____  __ _ _ __   ___  _ __
            // \ \ /\ / / _ \/ _` | '_ \ / _ \| '_ \
            //  \ V  V /  __/ (_| | |_) | (_) | | | |
            //   \_/\_/ \___|\__,_| .__/ \___/|_| |_|
            //   _ __  _   _ _ __ |_| |__   __ _ ___  ___
            // | '_ \| | | | '__/ __| '_ \ / _` / __|/ _ \
            // | |_) | |_| | | | (__| | | | (_| \__ \  __/
            // | .__/ \__,_|_|  \___|_| |_|\__,_|___/\___|
            // |_|

            QUERY(getCountOfMemberAttendancesInLastYear,
                " SELECT COUNT(*) as value "
                " FROM Attendance WHERE member_id = :memberId AND date >= DATE('now', '-1 year');",
                PARAM(oatpp::UInt32, memberId));

            QUERY(countDistinctAttendentMontsWithinLastYear,
                " SELECT COUNT(DISTINCT strftime('%m', date)) AS value "
                " FROM Attendance "
                " WHERE date >= DATE('now', '-1 year') "
                " AND member_id = :memberId; ",
                PARAM(oatpp::UInt32, memberId));
        };

#include OATPP_CODEGEN_END(DbClient)


    } // namespace component
} // namespace primus

#endif //DATABASE_CLIENT
```

DatabaseComponent.hpp

```
00001 #ifndef CRUD_DATABASECOMPONENT_HPP
00002 #define CRUD_DATABASECOMPONENT_HPP
00003
00004 #include "oatpp/core/macro/component.hpp"
00005
00006 #include "DatabaseClient.hpp"
00007 #include "filesystemHelper.hpp"
00008
00009 namespace primus
00010 {
00011     namespace component
00012     {
00013         //  ____        _        _                      ____
_
00014         // |  _ \  __ _| |_ __ _| |__   __ _ ___  ___ / ___|___  _ __ ___  _ __   ___
_ __   ___ _ __ | |_
00015         // | | | |/ _` | __/ _` | '_ \ / _` / __|/ _ \ |   / _ \| '_ ` _ \| '_ \ /
_ \| '_ \ / _ \ '_ \| __|
00016         // | |_| | (_| | || (_| | |_) | (_| \__ \  __/ |__| (_) | | | | | | |_) |
(_) | | | |  __/ | | | |_
00017         // |____/ \__,_|\__\__,_|_.__/ \__,_|___/\___|\____\___/|_| |_| |_| .__/
\___/|_| |_|\___|_| |_|\__|
00018         //                                                                |_|
00022         class DatabaseComponent {
00023         public:
00024             // Create database connection provider component
00025
OATPP_CREATE_COMPONENT(std::shared_ptr<oatpp::provider::Provider<oatpp::sqlite::Connec
tion>>, dbConnectionProvider)([] {
00026
00027                 /* Create database-specific ConnectionProvider */
00028                 auto connectionProvider =
std::make_shared<oatpp::sqlite::ConnectionProvider>(DATABASE_FILE);
00029
00030                 /* Create database-specific ConnectionPool */
00031                 return
oatpp::sqlite::ConnectionPool::createShared(connectionProvider,
00032                     10 /* max-connections */,
00033                     std::chrono::seconds(5) /* connection TTL */);
00034
00035                 }());
00036
00037             // Create database client
00038             OATPP_CREATE_COMPONENT(std::shared_ptr<DatabaseClient>, database)([]
{
00039
00040                 /* Get database ConnectionProvider component */
00041
OATPP_COMPONENT(std::shared_ptr<oatpp::provider::Provider<oatpp::sqlite::Connection>>,
connectionProvider);
00042
00043                 /* Create database-specific Executor */
00044                 auto executor =
std::make_shared<oatpp::sqlite::Executor>(connectionProvider);
00045
00046                 /* Create MyClient database client */
00047                 return std::make_shared<DatabaseClient>(executor);
00048
00049                 }());
00050
00051         };
00052
00053     } //namespace component
00054 } // namespace primus
00055
00056 #endif //CRUD_DATABASECOMPONENT_HPP
```

BooleanDto.hpp

```
00001 #ifndef BOOLEANDTO_HPP
00002 #define BOOLEANDTO_HPP
00003
00004 #include "oatpp/core/Types.hpp"
00005 #include "oatpp/core/macro/codegen.hpp"
00006
00007 namespace primus
00008 {
00009     namespace dto
00010     {
00011
00012 #include OATPP_CODEGEN_BEGIN(DTO)
00013         //  ____                  _                ____  _
00014         // | __ )  ___   ___ | | ___  __ _ _ __ |  _ \| |_ ___
00015         // |  _ \ / _ \ / _ \| |/ _ \/ _` | '_ \| | | | __/ _ \
00016         // | |_) | (_) | (_) | |  __/ (_| | | | | |_| | || (_) |
00017         // |____/ \___/ \___/|_|\___|\__,_|_| |_|____/ \__\___/
00021         class BooleanDto : public oatpp::DTO
00022         {
00023
00024             DTO_INIT(BooleanDto, DTO);
00025
00026             DTO_FIELD_INFO(value) {
00027                 info->description = "Boolean value";
00028             }
00029             DTO_FIELD(oatpp::Boolean, value);
00030
00031         };
00032
00033 #include OATPP_CODEGEN_END(DTO)
00034
00035     } // namespace dto
00036 } // namespace primus
00037
00038 #endif // BOOLEANDTO_HPP
```

DatabaseDtos.hpp

```
00001 #ifndef DATABASEDTOS_HPP
00002 #define DATABASEDTOS_HPP
00003
00004 #include "oatpp/core/Types.hpp"
00005 #include "oatpp/core/macro/codegen.hpp"
00006
00007 #include OATPP_CODEGEN_BEGIN(DTO)
00008
00009 namespace primus
00010 {
00011     namespace dto
00012     {
00013         namespace database
00014         {
00015             //      _                 _                          ____  _
00016             //     / \   __| | __| |_ __ ___  ___ ___|  _ \| |_ ___
00017             //    / _ \ / _` |/ _` | '__/ _ \/ __/ __| | | | __/ _ \
00018             //   / ___ \ (_| | (_| | | |  __/\__ \__ \ |_| | || (_) |
00019             //  /_/   \_\__,_|\__,_|_|  \___||___/___/____/ \__\___/
00020             class AddressDto : public oatpp::DTO
00021             {
00025                 DTO_INIT(AddressDto, DTO /* extends */)
00026
00027                     DTO_FIELD_INFO(id) {
00028                     info->description = "Unique identifier for the address";
00029                 }
00030                 DTO_FIELD(oatpp::UInt32, id);
00031
00032                 DTO_FIELD_INFO(postalCode) {
00033                     info->description = "ZIP code of the address";
00034                 }
00035                 DTO_FIELD(oatpp::String, postalCode);
00036
00037                 DTO_FIELD_INFO(city) {
00038                     info->description = "City of the address";
00039                 }
00040                 DTO_FIELD(oatpp::String, city);
00041
00042                 DTO_FIELD_INFO(country) {
00043                     info->description = "State of the address";
00044                 }
00045                 DTO_FIELD(oatpp::String, country);
00046
00047                 DTO_FIELD_INFO(houseNumber) {
00048                     info->description = "House number of the address";
00049                 }
00050                 DTO_FIELD(oatpp::UInt32, houseNumber);
00051
00052                 DTO_FIELD_INFO(street) {
00053                     info->description = "Street of the address";
00054                 }
00055                 DTO_FIELD(oatpp::String, street);
00056
00057             };
00058
00059             //  ____        _       ____  _
00060             // |  _ \  __ _| |_ ___|  _ \| |_ ___
00061             // | | | |/ _` | __/ _ \ | | | __/ _ \
00062             // | |_| | (_| | ||  __/ |_| | || (_) |
00063             // |____/ \__,_|\__\___|____/ \__\___/
00067             class DateDto : public oatpp::DTO
00068             {
00069
00070                 DTO_INIT(DateDto, DTO)
00071
00072                     DTO_FIELD(oatpp::String, date);
00073             };
00074
00075             //  ____                                 _                          _   ____  _
00076             // |  _ \  ___ _ __   __ _ _ __ | |_ _ __ ___   ___ _ __ | |_|  _ \| |_
___
00077             // | | | |/ _ \ '_ \ / _` | '__| __| '_ ` _ \ / _ \ '_ \| __| | | | __/
_ \
```

```
00078             // | |_| |  __/ |_) | (_| | |  | |_| | | | | |  __/ | | | |_| |_| | ||
(_) |
00079             // |____/ \___| .__/ \__,_|_|   \__|_| |_| |_|\___|_| |_|\__|____/
\__\___/
00080             //            |_|
00084             class DepartmentDto : public oatpp::DTO
00085             {
00086
00087                 DTO_INIT(DepartmentDto, DTO /* extends */)
00088
00089                     DTO_FIELD_INFO(id) {
00090                     info->description = "Unique identifier for each department";
00091                 }
00092                 DTO_FIELD(oatpp::UInt32, id);
00093
00094                 DTO_FIELD_INFO(name) {
00095                     info->description = "Name of the department";
00096                 }
00097                 DTO_FIELD(oatpp::String, name);
00098
00099             };
00100
00101             //  __  __                 _                  ____  _
00102             // |  \/  | ___ _ __ ___ | |__   ___ _ __|  _ \| |_ ___
00103             // | |\/| |/ _ \ '_ ` _ \| '_ \ / _ \ '__| | | | __/ _ \
00104             // | |  | |  __/ | | | | | |_) |  __/ |  | |_| | || (_) |
00105             // |_|  |_|\___|_| |_| |_|_.__/ \___|_|  |____/ \__\___/
00109             class MemberDto : public oatpp::DTO
00110             {
00111                 DTO_INIT(MemberDto, DTO /* extends */);
00112
00113                 DTO_FIELD_INFO(id) {
00114                     info->description = "Unique identifier for each member";
00115                 }
00116                 DTO_FIELD(oatpp::UInt32, id);
00117
00118                 DTO_FIELD_INFO(firstName) {
00119                     info->description = "First name of the member";
00120                 }
00121                 DTO_FIELD(oatpp::String, firstName);
00122
00123                 DTO_FIELD_INFO(lastName) {
00124                     info->description = "Last name of the member";
00125                 }
00126                 DTO_FIELD(oatpp::String, lastName);
00127
00128                 DTO_FIELD_INFO(email) {
00129                     info->description = "Email address of the member";
00130                 }
00131                 DTO_FIELD(oatpp::String, email);
00132
00133                 DTO_FIELD_INFO(phoneNumber) {
00134                     info->description = "Phone number of the member";
00135                 }
00136                 DTO FIELD(oatpp::String, phoneNumber);
00137
00138                 DTO_FIELD_INFO(birthDate) {
00139                     info->description = "Birth date of the member";
00140                 }
00141                 DTO_FIELD(oatpp::String, birthDate);
00142
00143                 DTO_FIELD_INFO(createDate) {
00144                     info->description = "Date when the member was created";
00145                 }
00146                 DTO_FIELD(oatpp::String, createDate);
00147
00148                 DTO FIELD INFO(notes) {
00149                     info->description = "Additional notes about the member";
00150                 }
00151                 DTO_FIELD(oatpp::String, notes);
00152
00153                 DTO FIELD INFO(active) {
00154                     info->description = "Whether or not the member is activated";
00155                 }
00156                 DTO_FIELD(oatpp::Boolean, active) = true;
00157
00158             };
```

```
00159 #include OATPP_CODEGEN_END(DTO)
00160         } // namespace database
00161     } // namespace dto
00162 } // namespace primus
00163 
00164 #endif // DATABASEDDTOS_HPP
```

Int32Dto.hpp

```
00001 #ifndef COUNTDTO_HPP
00002 #define COUNTDTO_HPP
00003
00004 #include "oatpp/core/Types.hpp"
00005 #include "oatpp/core/macro/codegen.hpp"
00006
00007 namespace primus
00008 {
00009     namespace dto
00010     {
00011
00012 #include OATPP_CODEGEN_BEGIN(DTO)
00013         //   _    _ ___          _   _____ _____ ___
00014         // | | | |_ _|_ __ | |_|___ /___ \|  _ \|_ ___|
00015         // | | | || || '_ \| __| |_ \ __) | | | | __/ _ \
00016         // | |_| || || | | | |_ ___) / __/| |_| | || (_) |
00017         //  \___/|___|_| |_|\__|____/_____|____/ \__\___/
00021         class UInt32Dto : public oatpp::DTO
00022         {
00023
00024             DTO_INIT(UInt32Dto, DTO);
00025
00026             DTO_FIELD_INFO(value) {
00027                 info->description = "General value of a single unsigned int";
00028             }
00029             DTO_FIELD(oatpp::UInt32, value);
00030
00031         };
00032         //  ___          _   _____ _____ ___
00033         // |_ _|_ __ | |_|___ /___ \|  _ \|_ ___|
00034         //  | || '_ \| __| |_ \ __) | | | | __/ _ \
00035         //  | || | | | |_ ___) / __/| |_| | || (_) |
00036         // |___|_| |_|\__|____/_____|____/ \__\___/
00040         class Int32Dto : public oatpp::DTO
00041         {
00042
00043             DTO_INIT(Int32Dto, DTO);
00044
00045             DTO_FIELD_INFO(value) {
00046                 info->description = "General value of a single signed int";
00047             }
00048             DTO_FIELD(oatpp::Int32, value);
00049
00050         };
00051
00052 #include OATPP_CODEGEN_END(DTO)
00053
00054     } // namespace dto
00055 } // namespace primus
00056
00057 #endif // COUNTDTO_HPP
```

PageDto.hpp

```
00001 #ifndef PAGEDTO_HPP
00002 #define PAGEDTO_HPP
00003
00004 #include "oatpp/core/Types.hpp"
00005 #include "oatpp/core/macro/codegen.hpp"
00006 #include "DatabaseDtos.hpp"
00007
00008 namespace primus
00009 {
00010     namespace dto
00011     {
00012 #include OATPP_CODEGEN_BEGIN(DTO)
00013         //  ____                  ____  _
00014         // |  _ \ __ _  __ _  ___|  _ \| |_ ___
00015         // | |_) / _` |/ _` |/ _ \ | | | __/ _ \
00016         // |  __/ (_| | (_| |  __/ |_| | || (_) |
00017         // |_|   \__,_|\__, |\___|____/ \__\___/
00018         //             |___/
00019         template<class T>
00020         class PageDto : public oatpp::DTO {
00021
00022             DTO_INIT(PageDto, DTO);
00023
00024             DTO_FIELD_INFO(offset) {
00025                 info->description = "Offset value for pagination";
00026             }
00027             DTO_FIELD(UInt32, offset);
00028
00029             DTO_FIELD_INFO(limit) {
00030                 info->description = "Limit value for pagination";
00031             }
00032             DTO_FIELD(UInt32, limit);
00033
00034             DTO_FIELD_INFO(count) {
00035                 info->description = "Total count of items";
00036             }
00037             DTO_FIELD(UInt32, count);
00038
00039             DTO_FIELD_INFO(items) {
00040                 info->description = "List of items";
00041             }
00042             DTO_FIELD(Vector<T>, items);
00043
00044         };
00045         //  __  __                 _                 ____                  ____  _
00046         // |  \/  | ___ _ __ ___ | |__   ___ _ __|  _ \ __ _  __ _  ___|  _ \| |_
___
00047         // | |\/| |/ _ \ '_ ` _ \| '_ \ / _ \ '__| |_) / _` |/ _` |/ _ \ | | | __/
_ \
00048         // | |  | |  __/ | | | | | |_) |  __/ |  |  __/ (_| | (_| |  __/ |_| | ||
(_) |
00049         // |_|  |_|\___|_| |_| |_|_.__/ \___|_|  |_|   \__,_|\__, |\___|____/
\__\___/
00050         //                                                   |___/
00051         class MemberPageDto : public
PageDto<oatpp::Object<primus::dto::database::MemberDto>> {
00052
00053             DTO_INIT(MemberPageDto,
PageDto<oatpp::Object<primus::dto::database::MemberDto>>)
00054
00055         };
00056 #include OATPP_CODEGEN_END(DTO)
00057
00058     } // namespace dto
00059 } // namespace primus
00060
00061 #endif // PAGEDTO_HPP
```

StatusDto.hpp

```
00001 #ifndef STATUSDTO_HPP
00002 #define STATUSDTO_HPP
00003
00004 #include "oatpp/core/macro/codegen.hpp"
00005 #include "oatpp/core/Types.hpp"
00006
00007 namespace primus
00008 {
00009     namespace dto
00010     {
00011 #include OATPP CODEGEN BEGIN(DTO)
00012         //  ____  _          _                ____  _
00013         // / ___|| |_ __ _ _| |_ _   _ ___   |  _ \| |_ ___
00014         // \___ \| __/ _` | __| | | / __|  | | | | __/ _ \
00015         //  ___) | || (_| | |_| |_| \__ \  |_| | | || (_) |
00016         // |____/ \__\__,_|\__|\__,_|___/____/ \__\___/
00017         class StatusDto : public oatpp::DTO
00018         {
00019             DTO_INIT(StatusDto, DTO);
00020
00021             DTO_FIELD_INFO(status) {
00022                 info->description = "Short status text";
00023             }
00024             DTO_FIELD(String, status);
00025
00026             DTO_FIELD_INFO(code) {
00027                 info->description = "Status code";
00028             }
00029             DTO_FIELD(Int32, code);
00030
00031             DTO_FIELD_INFO(message) {
00032                 info->description = "Verbose message";
00033             }
00034             DTO_FIELD(String, message);
00035
00036         };
00037 #include OATPP_CODEGEN_END(DTO)
00038
00039     } // namespace dto
00040 } // namespace primus
00041 #endif // STATUSDTO_HPP
```

filesystemHelper.hpp

```
00001 #ifndef FILESYSTEM_CLIENT_H
00002 #define FILESYSTEM_CLIENT_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <string>
00007
00008 #ifdef _WIN32
00009 #include <Windows.h>
00010 #include <direct.h>
00011 #elif   linux
00012 #include <sys/stat.h>
00013 #include <unistd.h>
00014 #endif
00015
00016 namespace primus {
00017     namespace component {
00018         namespace filesystem {
00019             // Not fully functional. May be implemented later on
00020             //         std::string getExecutablePath()
00021             //         {
00022             //             char buffer[FILENAME_MAX];
00023             // #ifdef _WIN32
00024             //             GetModuleFileNameA(NULL, buffer, FILENAME_MAX);
00025             // #elif __linux__
00026             //             ssize_t len = readlink("/proc/self/exe", buffer,
sizeof(buffer) - 1);
00027             //             if (len != -1) {
00028             //                 buffer[len] = '\0';
00029             //             }
00030             // #endif
00031             //             return std::string(buffer);
00032             //         }
00033             //
00034             //         bool createDirectory(const std::string& path)
00035             //         {
00036             // #ifdef _WIN32
00037             //             if (_mkdir(path.c_str()) == 0) {
00038             // #elif __linux__
00039             //             if (mkdir(path.c_str(), 0777) == 0) {
00040             // #endif
00041             //                 return true;
00042             //             }
00043             //             else {
00044             //                 return false;
00045             //             }
00046             //         }
00047             //
00048             //         bool createFile(const std::string & fileName, const
std::string & content)
00049             //         {
00050             //             std::ofstream file(fileName);
00051             //             if (file.is_open()) {
00052             //                 file << content;
00053             //                 file.close();
00054             //                 return true;
00055             //             }
00056             //             else {
00057             //                 return false;
00058             //             }
00059             //         }
00060         } // filesystem
00061     } // namespace component
00062 } // Namespace primus
00063 #endif // FILESYSTEM_CLIENT_H
```

asserts.hpp

```
00001 #ifndef PRIMUSASSERTS_HPP
00002 #define PRIMUSASSERTS_HPP
00003
00004 #include "oatpp/core/Types.hpp"
00005 #include "oatpp/core/macro/component.hpp"
00006
00007 #include "dto/StatusDto.hpp"
00008 #include "dto/PageDto.hpp"
00009 #include "dto/Int32Dto.hpp"
00010 #include "dto/BooleanDto.hpp"
00011 #include "general/constants.hpp"
00012 #include "dto/DatabaseDtos.hpp"
00013
00014 namespace primus
00015 {
00016     namespace assert
00017     {
00018         oatpp::Object<primus::dto::StatusDto> assertMemberExists(const
oatpp::UInt32 memberId)
00019         {
00020             OATPP_COMPONENT(std::shared_ptr<primus::component::DatabaseClient>,
m_database);
00021
00022             oatpp::Object<primus::dto::StatusDto> ret =
primus::dto::StatusDto::createShared();
00023
00024             auto dbResult = m_database->getMemberById(memberId);
00025
00026             ret->code = 500;
00027             ret->message = "Unknown error";
00028             ret->status = "During check for wheather or not a member exists an unknown
error accourd";
00029
00030             if (!dbResult->isSuccess())
00031             {
00032                 ret->code = 500;
00033                 ret->message = dbResult->getErrorMessage();
00034                 ret->status = "Failed to ask for member at database";
00035             }
00036             else
00037             {
00038                 auto member =
dbResult->fetch<oatpp::Vector<oatpp::Object<primus::dto::database::MemberDto>>>();
00039
00040                 if (member->size() == 0)
00041                 {
00042                     ret->code = 404;
00043                     ret->message = "Database request was successfully executed. The
retrieved data did not include a member";
00044                     ret->status = "Member could not be found";
00045                 }
00046                 else if(member->size() > 1)
00047                 {
00048                     OATPP_LOGE("Primus Assertion: ", "CRITICAL DATABASE ERROR: MORE
THAN ONE USER WITH ID %d", memberId.operator v_uint32());
00049                     ret->code = 500;
00050                     ret->message = "During check for wheather a user exists, the
retrieved data contained 2 seperate MemberDtos";
00051                     ret->status = "CRITICAL DATABASE ERROR: MORE THAN ONE USER";
00052                 }
00053                 else
00054                 {
00055                     ret->code = 200;
00056                     ret->message = "OK";
00057                     ret->status = "Member was found";
00058                 }
00059             }
00060             return ret;
00061         }
00062     }   // Namespace asserts
00063 }   // Namespace primus
00064
00065 #endif // PRIMUSASSERTS_HPP
```

constants.hpp

```
00001 #ifndef PRIMUSCONSTANTS_HPP
00002 #define PRIMUSCONSTANTS_HPP
00003
00004 #include <cstddef>
00005
00006 namespace primus
00007 {
00008     namespace constants
00009     {
00010         const std::size_t logNameLength = 20;
00011         const std::size t logSeperationLength = 60;
00012
00013         namespace pricing
00014         {
00015             enum DepartmentPrices
00016             {
00017                 Bogenschiessen = 8,
00018                 Luftdruck = 10,
00019                 Schusswaffen = 15,
00020                 Multiple = 20,
00021                 None = 0
00022             };
00023         }
00024
00025         namespace main
00026         {
00027             const char logName[logNameLength]             = "Main initialization";
00028             const char logSeperation[logSeperationLength] =
"------------------------------";
00029         } // Namespace main
00030
00031         namespace databaseclient
00032         {
00033             const char logName[logNameLength] = "DatabaseClient     ";
00034             const char logSeperation[logSeperationLength] =
"------------------------------";
00035         }
00036
00037         namespace apicontroller
00038         {
00039             namespace static_endpoint
00040             {
00041                     // Name and seperation while logging
00042                     const char logName[logNameLength]             =
"StaticController   ";
00043                     const char logSeperation[logSeperationLength] =
"------------------------";
00044             } // Namespace static_endpoint
00045
00046             namespace member_endpoint
00047             {
00048                     // Name and seperation while logging
00049                     const char logName[logNameLength]             =
"MemberController   ";
00050                     const char logSeperation[logSeperationLength] =
"------------------------";
00051             } // Namespace Member
00052         } // Namespace ApiController
00053     } // Namespace constants
00054 } // Namespace Primus
00055 #endif // PRIMUSCONSTANTS_HPP
```

SwaggerComponent.hpp

```
00001
00002 #ifndef SwaggerComponent_hpp
00003 #define SwaggerComponent_hpp
00004
00005 #include "oatpp-swagger/Model.hpp"
00006 #include "oatpp-swagger/Resources.hpp"
00007 #include "oatpp/core/macro/component.hpp"
00008
00009 namespace primus
00010 {
00011     namespace component
00012     {
00017         class SwaggerComponent {
00018         public:
00019
00023
OATPP_CREATE_COMPONENT(std::shared_ptr<oatpp::swagger::DocumentInfo>,
swaggerDocumentInfo)([] {
00024
00025                 oatpp::swagger::DocumentInfo::Builder builder;
00026
00027                 builder
00028                     .setTitle("Primus")
00029                     .setDescription("Swagger documentation for the primus
project")
00030                     .setVersion("1.0")
00031                     .setContactName("Sascha Meissner")
00032                     .setContactUrl("https://thesascham.github.io/Website/")
00033
00034                     .setLicenseName("Apache License, Version 2.0")
00035                     .setLicenseUrl("http://www.apache.org/licenses/LICENSE-2.0")
00036
00037                     .addServer("http://localhost:8000", "server on localhost");
00038
00039                 return builder.build();
00040
00041                 }());
00042
00046             OATPP_CREATE_COMPONENT(std::shared_ptr<oatpp::swagger::Resources>,
swaggerResources)([] {
00047                 // Make sure to specify correct full path to oatpp-swagger/res folder
!!!
00048                 return
oatpp::swagger::Resources::loadResources(OATPP_SWAGGER_RES_PATH);
00049                 }());
00050
00051         };
00052     } //namespace component
00053 } // namespace primus
00054
00055 #endif /* SwaggerComponent_hpp */
```

Index

INDEX