Samuel Rengifo

Client	Mayor's Office of Cali
User	COP Members Cali
Context of the problem	The 16th edition of the United Nations Framework Convention on Climate Change, one of the most important in the world, will be held in the city of Cali. Therefore, the Mayor of the city of Cali, requires a program that allows knowledge and navigation of both sites with biological diversity and communities that care for those places.
Functional requirements	FR0: Register community FR1: Register a place FR2: Register a product FR3: Delete a product FR4: Add species to a place FR5: Update species data in a place FR6: Query information about a place FR7: Query the information of the communities of a department FR8: Query communities with major problems FR9: Query the name of the place with the highest number of species FR10: Query the three largest places per square kilometer
Non-functional requirements	NFR0: Registered data must be protected NFR1: Intuitive interface NFR2: Cross-platform accessibility NFR3: The program must have documentation to facilitate future updates

Name or identifier	FR0: Register community		
Summary	This functional requirement will give the user the possibility to register a community filling some required values.		
Input	Input name	Datatype	Selection condition or repetition
	communityName	String	This string does not have any specific condition, but it must

			contain max 50 characters.
	communityType	String	This selection should be one of the three following options: [1] Afrocolombian, [2] Indigenous or [3] Raizal.
	communityBossName	String	This string does not have any specific condition, but it must contain max 50 characters.
	communityBossPhoneNumber	int	This value should have exactly 10 digits.
	communityMajorProblem	String	This selection should be one of the four following options: [1] Lack of hospital, [2] Lack of school, [3] Lack of drinking water, or [4] Lack of access to basic food.
	communityProduct	Product	Product must be already registered, to add it to a certain community.
	communityAmountInhabitants	int	This value cannot be negative.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition
	successfulRegistrationMessage	String	The String message will inform the user that the process has been completed successfully.
	errorRegistrationMessage	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR1: Register a place

Summary	This functional requirement will give the user the possibility to register a place filling some required values.		
Input	Input name	Datatype	Selection condition or repetition
	placeName	String	This string does not have any specific condition, but it must contain max 50 characters.
	placeDepartment	String	This selection should be one of the four following options: [1] Choco, [2] Valle, [3] Cauca, or [4] Narino.
	amountOfSquareKm	int	This value cannot be negative.
	placeType	String	This selection should be one of the three following options: [1] Protected, [2] National Park, or [3] Private.
	placeOpeningDate	Date	This selection value should be imputed in the following format: DAY/MONTH/YEAR
	placePhoto	String	This String value should refer to a valid URL direction, which contains the image to save.
	placeCommunity	String	This value should be a community that has been created before, so in that order, this String, should just link them.
	placeFinancialResourcesRequired	double	This value cannot be negative.
	placeLivingSpecie	Specie	Specie must be already registered, to add it to a certain place.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition

	successfulRegistration Message	String	The String message will inform the user that the process has been completed successfully.
	errorRegistrationMess age	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR2: Register a product		
Summary	This functional requirement will give the user the possibility to register 20 products max for each community, just filling some required values.		
Input	Input name	Datatype	Selection condition or repetition
	productName	String	This string does not have any specific condition, but it must contain max 50 characters.
	productNaturalPercent age	double	This value cannot be negative.
	productType	String	This selection should be one of the two following options: [1] Food, or [2] Craft.
	productHandmade	String	This selection must be YES or NO.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition
	successfulRegistration Message	String	The String message will inform the user that the process has been completed successfully.
	errorRegistrationMess age	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR3: Delete a product		
Summary	This functional requirement will enable the user to delete a product that has been already registered.		
Input	Input name	Datatype	Selection condition or repetition
	communityId	Community	Community must be already registered and linked to a product.
	productId	Product	Product must be already registered, to delete it.
Result or postcondition	The imputed values will work as identifiers on the system database, if they are correct, the selected product will be deleted, if they are not, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	successfulProcessMessage	String	The String message will inform the user that the process has been completed successfully.
	errorProcessMessage	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR4: Add species to a place		
Summary	This functional requirement will enable the user to create a specie and add it to a place.		
Input	Input name	Datatype	Selection condition or repetition
	placeId	Place	Place must be already registered to link it to a specie.
	specieName	String	This string does not have any specific condition, but it must contain max 50 characters.
	specieBiodiversityType	String	This selection should be one of the two following options: [1] Flora, or [2] Fauna.

	speciePhoto	String	This String value should refer to a valid URL direction, which contains the image to save.
	specieLiving	int	This value cannot be negative.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition
	successfulRegistration Message	String	The String message will inform the user that the process has been completed successfully.
	errorRegistrationMess age	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR5: Update species data in a place		
Summary	This functional requirement will enable the user to update a specie data in a place.		
Input	Input name	Datatype	Selection condition or repetition
	placeId	Place	Place must be already registered.
	newSpecieName	String	This string does not have any specific condition, but it must contain max 50 characters.
	newSpecieBiodiversity Type	String	This selection should be one of the two following options: [1] Flora, or [2] Fauna.
	newSpeciePhoto	String	This String value should refer to a valid URL direction, which contains the image to save.
	newSpecieLiving	int	This value cannot be negative.

Result or postcondition	The imputed values are going to replace the past values stored on the system database if the conditions are correct, if they are not, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	successfulProcessMessage	String	The String message will inform the user that the process has been completed successfully.
	errorProcessMessage	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR6: Query information about a place		
Summary	This functional requirement will show to the user all the information about a specific place.		
Input	Input name	Datatype	Selection condition or repetition
	placeId	Place	Place must be already registered.
Result or postcondition	The imputed value will work as an identifier on the system database, if it is correct, all the information about the selected place will be displayed, if the identifier is not valid, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	placeName	String	All these values are going to be displayed on the console as they were originally imputed.
	placeDepartment	String	
	amountOfSquareKm	int	
	placeType	String	
	placeOpeningDate	Date	
	placePhoto	String	
	placeCommunity	String	
	placeFinancialResourcesRequired	double	
	placeLivingSpecie	Specie	
	errorProcessMessage	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR7: Query the information of the communities of a department		
Summary	This functional requirement will show to the user all the information about all the communities in a specific department.		
Input	Input name	Datatype	Selection condition or repetition
	placeDepartment	String	This String must be already registered to a place.
Result or postcondition	The imputed value will work as an identifier on the system database, if it is correct, all the information about the communities in the selected department will be displayed, if the identifier is not valid, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	communityName	String	All these values are going to be displayed on the console as they were originally imputed.
	communityType	String	
	communityBossName	String	
	communityBossPhone Number	int	
	communityMajorProbl em	String	
	communityProduct	Product	
	communityAmountInh abitants	int	
	errorProcessMessage	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR8: Query communities with major problems		
Summary	This functional requirement will show to the user all the communities that match with the selected major problem.		
Input	Input name	Datatype	Selection condition or repetition
	communityMajorProbl em	String	This selection should be one of the four following options: [1] Lack of hospital, or [2] Lack of school.
Result or postcondition	The imputed value will work as an identifier on the system database, if it is correct, all the information about the communities that match with the major problem typed, will be displayed, if the identifier is not valid, an error message will be produced.		

Output	Output name	Datatype	Selection condition or repetition
	communityName	String	All these values are going to be displayed on the console as they were originally imputed.
	communityType	String	
	communityBossName	String	
	communityBossPhone Number	int	
	communityMajorProbl em	String	
	communityProduct	Product	
	communityAmountInh abitants	int	
	errorProcessMessage	String	The String message will inform the user that something must have gone wrong during the process.

Name or identifier	FR9: Query the name of the place with the highest number of species		
Summary	This functional requirement will show to the user, the place with the highest number of species.		
Input	Input name	Datatype	Selection condition or repetition
	N/A		
Result or postcondition	By executing a method, the system will search between the registered places, which is the place with the highest number of species, and finally it will show on screen the name of the place that applies to the condition.		
Output	Output name	Datatype	Selection condition or repetition
	placeName	String	This String value will be displayed according to the condition.

Name or identifier	FR10: Query the three largest places per square kilometer		
Summary	This functional requirement will show to the user, the three largest places per square kilometer.		
Input	Input name	Datatype	Selection condition or repetition
	N/A		

Result or postcondition	By executing a method, the system will search between the registered places, which are the three largest places per square kilometer, and finally it will show on screen the name of the three places that apply to the condition.		
Output	Output name	Datatype	Selection condition or repetition
	placeNameOne	String	The Strings values will be displayed according to the applied condition.
	placeNameTwo		
	placeNameThree		

Functional Requirement	**Class Name**	**Method name**
FR0 Register community	Class CopSystem	+registerCommunity() : void
	Class Controller	+addCommunity(communityName : String, communityType : String, communityBossName : String, communityBossPhoneNumber : int, communityMajorProblem : String, communityProduct : Product, communityAmountInhabitants : int) : boolean
	Class Community	+Community(communityName : String, communityType : String, communityBossName : String, communityBossPhoneNumber : int, communityMajorProblem : String, communityProduct : Product, communityAmountInhabitants : int)
FR1 Register a place	Class CopSystem	+registerPlace() : void
	Class Controller	+addPlace(placeName : String, placeDepartment : String, amountOfSquareKm : int, placeType : String, placeOpeningDate : Date, placePhoto : String, placeCommunity : String, placeFinancialResourcesRequired : double, placeLivingSpecie : Specie) : boolean

	Class Place	+Place(placeName : String, placeDepartment : String, amountOfSquareKm : int, placeType : String, placeOpeningDate : Date, placePhoto : String, placeCommunity : String, placeFinancialResourcesRequired : double, placeLivingSpecie : Specie)
FR2 Register a product	Class CopSystem	+registerProduct() : void
	Class Controller	+addProduct(productName : String, productNaturalPercentage : double, productType : String, productHandmade : String) : boolean
	Class Community	+addProduct(productId : Product) : boolean
	Class Product	+Product(productName : String, productNaturalPercentage : double, productType : String, productHandmade : String)
FR3 Delete a product	Class CopSystem	+deleteProduct() : void
	Class Controller	+eraseProduct(communityId : Community, productId : Product) : boolean
	Class Community	+eraseProduct(productId : Product) : boolean
FR4 Add species to a place	Class CopSystem	+addSpecieToPlace() : void
	Class Controller	+addSpecieToPlace(placeId : Place, specieName : String, specieBiodiversityType : String, speciePhoto : String, specieLiving : int) : boolean
	Class Place	+addSpecieToPlace(specieId : Specie) : boolean
	Class Specie	+Specie(specieName : String, specieBiodiversityType : String, speciePhoto : String, specieLivingAmount : int)

FR5 Update species data in a place	Class CopSystem	+updateSpecieToPlace() : void
	Class Controller	+setUpdatedSpecieToPlace(placeId : Place, newSpecieName : String, newSpecieBiodiversityType : String, newSpeciePhoto : String, newSpecieLivingAmount : int) : boolean
	Class Place	+searchSpecie(specieName : String) : Specie
	Class Specie	+setUpdatedSpecie(newSpecieName : String, newSpecieBiodiversityType : String, newSpeciePhoto : String, newSpecieLivingAmount : int) : void
FR6 Query information about a place	Class CopSystem	+showPlace() : void
	Class Controller	+showPlace(placeId : Place) : String
	Class Place	+getPlaceName() : String
		+getPlaceDepartment() : String
		+getAmountOfSquareKm() : int
		+getPlaceType() : String
		+getPlaceOpeningDate() : Date
		+getPlacePhoto() : String
		+getPlaceCommunity() : String
		+getPlaceFinancialResourcesRequired() : double
		+getPlaceLivingSpecie() : Specie
FR7 Query the information of the communities of a department	Class CopSystem	+showDepartmentCommunity() : void
	Class Controller	+showDepartmentCommunity(placeDepartment : String) : String
	Class Place	+getCommunityByDepartment() : Community []

	Class Community	+getCommunityName() : String
		+getCommunityType() : String
		+getCommunityBossName() : String
		+getCommunityBossPhoneNumber() : int
		+getCommunityMajorProblem() : String
		+getCommunityProduct() : Product
		+getCommunityAmountInhabitants() : int
FR8 Query communities with major problems	Class CopSystem	+showCommunityWithMajorProblems() : void
	Class Controller	+showCommunityWithMajorProblems(communityMajorProblem : String) : String
	Class Community	+getCommunityName() : String
		+getCommunityType() : String
		+getCommunityBossName() : String
		+getCommunityBossPhoneNumber() : int
		+getCommunityMajorProblem() : String
		+getCommunityProduct() : Product
		+getCommunityAmountInhabitants() : int
FR9 Query the name of the place with the highest number of species	Class CopSystem	+showPlaceNameWithMostSpecies() : void
	Class Controller	+showPlaceNameWithMostSpecies() : String
	Class Place	+getSpecie() : Specie []
	Class Specie	+getSpecieLivingAmount() : int

FR10 Query the three largest places per square kilometer	Class CopSystem	+showThreeBiggestPlacePerSquareKm() : void
	Class Controller	+showThreeBiggestPlacePerSquareKm() : String
	Class Place	+getAmountOfSquareKm() : int