



SMARTCONDO

Relatório do Trabalho Final de Curso

INGRID FRAGA MOTTÉ FERREIRA

SMARTCONDO

Relatório do Trabalho Final de Curso

Relatório do Trabalho Final de Curso da Licenciatura
em Engenharia Informática, da Universidade Lusófona
de Humanidades e Tecnologias.

Orientador: Professor Pedro Freire

Universidade Lusófona de Humanidades e Tecnologias
Escola de Comunicação, Artes e Tecnologias da Informação

Lisboa

2011

Resumo

O presente projecto final de curso consiste no desenvolvimento de um sistema de gestão de condomínios, que tem por objectivo facilitar esta tarefa tanto aos condóminos como ao administrador. O projecto consiste na concepção e desenvolvimento de um sistema de informação para a Web (PC) que é desenvolvido na plataforma ASP.NET.

Palavras-Chave: Condomínio Residencial, Gestão, Condómino, Síndico, Administrador.

Abstract

This final project pretends to develop a management system of condominiums in order to facilitate the management of co-owners and the administrators. The project comprises the design and development of an information system for the Web (PC) and development in platform ASP.NET.

Key-words: Residential Condominium, Management, Condominium, Syndic, Administrator.

Índice

Introdução	7
1. Enquadramento	8
1.1. O mercado dos Condomínios Fechados	9
1.2. Apresentação do Projecto SmartCondo	10
1.2.1. O Administrador	10
1.2.2. O Síndico	11
1.2.3. O Condómino	11
2. Método – Desenvolvimento do Protótipo	13
2.1. Engenharia de Requisitos	13
2.2. Funcionalidades Implementadas no Sistema	14
2.2.1. Adicionar Novo Condomínio	14
2.2.2. Seleccionar Condomínio	14
2.2.3. Histórico do Condomínio	15
2.2.4. Editar Dados Condomínio	16
2.2.5. Inserir Área Privada	16
2.2.6. Cancelar Contracto de Condomínio	17
2.2.7. Registar Novo Morador	18
2.2.8. Editar Dados Morador	18
2.2.9. Listar Moradores	19
2.2.10. Remover Moradores	20
2.2.11. Lançar Despesa Condomínio	21
2.2.12. Calcular Cota do Condomínio	21
2.2.13. Visualizar Cotas Lançadas	22
2.2.14. Pagamentos Contas do Condomínio	23
2.2.15. Histórico do Pagamento de Contas do Condomínio	23

2.2.16. Pagamentos Morador	24
2.2.17. Histórico Pagamento Morador	25
2.2.18. Reserva de Espaço	26
2.2.19. Gestão de Reserva	26
2.2.20. Agenda Morador	27
2.2.21. Convocar Assembleia	28
2.2.22. Gerar Relatório com Lista Moradores	29
2.2.23. Autenticação	30
2.2.24. Recuperação de senha	30
2.2.25. Logout	31
2.3. Modelo de Dados	32
2.3.1. Representação UML – Use Case	32
2.3.2. Representação UML – Diagrama de Classe	34
2.3.3. Descrição das Tabelas	35
2.3.3.1. Tabela Conta	35
2.3.3.2. Tabela CotaCondo	35
2.3.3.3. Tabela Detalhe_AreaPrivada	36
2.3.3.4. Tabela ItemAreaPrivada	36
2.3.3.5. Tabela Perfil	36
2.3.3.6. Tabelas PgtoCota	37
2.3.3.7. Tabelas Predio	37
2.3.3.8. Tabela Reserva	38
2.3.3.9. Tabela StatusConta	38
2.3.3.10. Tabela TipoConta	38
2.3.3.11. Tabela Utilizador	39
Conclusão	40
Bibliografia	41
Anexos	42

Índice de Figuras

Figura 1	Página Adicionar Novo Condomínio	14
Figura 2	Página Seleccionar Condomínio	15
Figura 3	Página Histórico do Condomínio	15
Figura 4	Página Editar Dados Condomínio	16
Figura 5	Página Inserir Área Privada	17
Figura 6	Página Cancelar Contracto do Condomínio	17
Figura 7	Página Registrar Novo Morador	18
Figura 8	Página Editar Dados Morador	19
Figura 9	Página Listar Moradores	20
Figura 10	Página Remover Moradores	20
Figura 11	Página Lançar Despesa Condomínio	21
Figura 12	Página Calcular Quota do Condomínio	22
Figura 13	Página Visualizar Quotas Lançadas	22
Figura 14	Página Pagamentos Contas do Condomínio	23
Figura 15	Página Histórico do Pagamento de Contas do Condomínio	24
Figura 16	Página Pagamentos Morador	25
Figura 17	Página Histórico Pagamento Morador	25
Figura 18	Página Reserva de Espaço	26
Figura 19	Página Gestão de Reserva	27
Figura 20	Página Agenda Morador	27
Figura 21	Página Convocar Assembleia	28
Figura 22	Email recebido pelo morador com a convocação da assembleia	28
Figura 23	Página Gerar Relatório com Lista de Moradores	29
Figura 24	Relatório Gerado em Formato PDF	29
Figura 25	Página Autenticação	30
Figura 26	Página Recuperação de Senha	31
Figura 27	Email de Recuperação de Senha	31

Introdução

O presente trabalho final de curso da licenciatura em Engenharia Informática, consiste no desenvolvimento de um projecto denominado *SmartCondo* para gestão de condomínios de habitação. Este projecto consiste numa aplicação comercializada avulso, que pode ser adquirida por qualquer condomínio residencial privado/fechado, sem que este tenha que recorrer a uma terceira empresa. SmartCondo possibilita a qualquer utilizador, mesmo que sem formação em administração e contabilidade possa gerir com facilidade o seu condomínio.

O protótipo funcional apresentado foi desenvolvido na ferramenta de desenvolvimento Visual Studio 2008, através da plataforma ASP.net e na linguagem de programação C#. Este protótipo tem implementado todas as funcionalidades que a aplicação final disponibilizaria, no entanto a comunicação online com entidades bancárias não está implementada.

Com este sistema o condómino, o síndico e o administrador terão acesso às transacções e despesas realizadas no seu condomínio. O administrador tem acesso ao servidor e pode inserir, editar ou eliminar informações, enquanto que o síndico e os condóminos apenas podem visualizar.

O Sistema é uma ferramenta de administração centralizada, que permite uma gestão online, de qualquer local, bastando apenas uma senha para acesso a toda estrutura disponível.

Esta aplicação permite gerir os pagamentos de em água, luz, funcionários, bem como visualizar o histórico de cada condómino, nomeadamente dívidas e histórico de pagamentos. Além da gestão da contabilidade o sistema permitirá ao gestor de condomínios fazer marcação de reuniões, apresentar actas das reuniões e enviar avisos por correio electrónico aos condóminos.

1. Enquadramento

O termo condomínio é frequentemente utilizado para definir o direito exercido pelos condóminos sobre as suas unidades privadas e sobre dependências edificadas de uso comum. Estes podem ser na forma horizontal ou vertical, residencial ou comercial.

Segundo Rita Raposo, os condomínios fechados (CFs) passaram a fazer parte do panorama espacial e social de muitas cidades e regiões do mundo. Terá sido a partir dos Estados Unidos da América que os CFs, ou *gated communities* (GCs), conforme a designação norte-americana, alcançaram grande parte do planeta. No que respeita a Portugal, é provável que se não tenha tratado total ou exactamente de um caso de ligação directa. A via Brasil pode ter sido particularmente importante. Certo é que, pelo menos desde os anos 1980, os CFs, também vulgarmente denominados de «privados», aportaram em território português.” (Raposo, 2008, 109). As Áreas Metropolitanas de Lisboa e do Porto, a par de algumas zonas turísticas do país, com destaque para o Algarve, constituíram aí o principal destino.

Para além de ocuparem um lugar próprio no *espaço* e *sociedade* contemporâneos e de terem despertado desde o final dos anos 1990 a constituição de um campo de pesquisa inteiramente novo, os CFs podem ser entendidos como o paradigma ou o reflexo sintético de alguns fenómenos, como transformações sociais e espaciais, a globalização, processos de reestruturação económica, uma nova estrutura social e uma nova relação entre classes ou grupos sociais; assinalam a crescente preocupação com a questão da segurança; reflectem mudanças culturais e o advento de novos estilos de vida; são signo do avanço da mercantilização e da racionalização da vida social; ilustram algumas das mais importantes transformações das esferas ideológicas e política; constituem, por último, eles próprios, uma das múltiplas (novas) paisagens que compõem o panorama cada vez mais fragmentado da metrópole contemporânea.

Segundo Paul Knox, os condomínios fechados podem, de facto, ser percebidos como uma das paisagens que preenchem e caracterizam de modo distintivo o espaço do nosso tempo. «Packaged landscapes» (Knox, 1992) foi a expressão que o autor utilizou para nomear estas paisagens, como centros comerciais, parques de escritórios, *outlets*, parques temáticos e CFs constituem apenas alguns exemplos.

Segundo Rita Raposo os CFs devem ser interpretados, simultaneamente, como uma forma de segregação única e como um produto imobiliário específico (Raposo, 2002 e 2003).

Enquanto forma de segregação ou de espacialização de desigualdades sociais, os CFs distinguem-se graças à associação única de dois traços: (1) recurso a barreiras físico-arquitectónicas; (2) carácter voluntário. Os CFs, reflectem um método específico de consagração espacial de distâncias sociais, o “policiamento arquitectónico” (Davis, 1990).

1.1. O mercado dos Condomínios Fechados

Segundo o Instituto Brasileiro de Geografia e Estatística, o mercado dos condomínios está com um crescimento promissor, principalmente no Brasil que está num momento de expansão e crescimento imobiliário. A expansão da classe média no Brasil, que agregou 35 milhões de pessoas nos últimos oito anos, ajuda a expandir a procura por crédito imobiliário, mas este processo está a decorrer de forma saudável. Estima-se que nos próximos dez anos o mercado imobiliário que hoje representa 5% do PIB passe a representar 15%. Por outro lado em Portugal, a tendência dos condomínios fechados também é crescente, no entanto, a outra escala. O mercado dos condomínios têm-se tornado de tal forma promissor, que já existem vários cursos específicos para gestão de condomínios.

A gestão dos condomínios também mudou de um perfil mais amador centralizado na figura do síndico para algo mais profissional com a figura do administrador de condomínios.

O tema ganhou tal relevância que algumas universidades brasileiras criaram o curso de gestão de condomínios.

Gerir condomínios não é uma tarefa simples, porque se existe áreas comuns a todos os condóminos como o “hall” de entrada, o elevador, a piscina, a sala de festas, estes devem contribuir para as despesas e é necessário que exista um administrador com competências necessárias para garantir que tudo funcione de forma adequada.

Actualmente, é cada vez mais evidente e necessário o recurso aos serviços de administração de condomínios. A legislação que regulamenta a Propriedade horizontal é vasta, específica e rigorosa. No entanto, a constante manutenção nos edifícios obriga a uma grande disponibilidade e empenho por parte dos condóminos, muitas vezes incompatível com a falta de tempo, característica do ritmo de vida actual.

É neste contexto que surge o Projecto *SmartCondo*, que pretende facilitar e agilizar a gestão de condomínios para os condóminos.

1.2. Apresentação do Projecto SmartCondo

Existe no mercado, tanto Português como Brasileiro, algumas empresas que prestam serviços de gestão de condomínios¹, no entanto SmartCondo surge como uma alternativa a estes serviços prestados por terceiros. SmartCondo é uma aplicação vendida avulso e instalável através de um CD. O condomínio adquire este sistema e selecciona entre os condóminos, um síndico e um administrador da aplicação.

SmartCondo é uma aplicação gerida por apenas um administrador, que é o único que tem acesso ao servidor e pode inserir, editar ou eliminar informações, por outro lado, tanto o síndico como os restantes condóminos acedem ao portal SmartCondo apenas *online*, mas não têm privilégios, apenas podem visualizar conteúdos, ou no caso do síndico pode também convocar assembleias e aprovar solicitações de reservas de áreas privadas do condomínio.

Cada utilizador (condómino, síndico ou administrador) tem um respectivo *nome de utilizador* e *código de acesso* para autenticação no Sistema.

1.2.1 O Administrador

Numa área de negócio como a da administração de condomínios, existe uma grande variedade de serviços e procedimentos que devem ser realizados com prazos e periodicidade específicos. Neste sentido o administrador deverá ter uma responsabilidade superior a um síndico e deverá ser da confiança dos moradores. Há cinco formas de eleger um novo administrador, através de eleição², por nomeação sucessiva³, a pagar pela administração⁴, nomeação pelo tribunal⁵ e finalmente o administrador “tapa-buracos”⁶.

¹ Empresas que prestam serviços de gestão de condomínios: <http://www.h24gestaodecondominios.com/>, <http://www.gestaodecondominios.pt/>, <http://www.charib.com/>

² Forma mais comum, que a lei prevê por defeito. Salvo disposição em contrário no regulamento ou por deliberação da assembleia de condóminos, o mandato do administrador é de um ano, renovável

³ Como nem sempre há candidatos, é frequente “distribuir o mal pelas aldeias”: o primeiro mandato pode ser exercido pelo proprietário da fracção A, o segundo pelo da B e assim sucessivamente.

⁴ Criar incentivos para quem desempenha o cargo pode ser motivador. A assembleia pode atribuir um salário ao administrador ou recorrer a uma empresa

⁵ Qualquer condómino pode pedir ao tribunal para nomear o administrador. Enquanto decorre o processo, as funções são desempenhadas a título provisório pelo proprietário com maior permissão (ou por ordem alfabética da fracção, em caso de empate).

De acordo com o artº1436 do Código Civil as funções do Administrador são: convocar a Assembleia de Condóminos; elaborar o orçamento de exploração do Edifício relativo ao ano/mês; efectuar e manter actualizado o seguro do edifício contra risco de incêndio; cobrar as receitas e efectuar as despesas comuns; exigir aos condóminos a sua parte nas despesas aprovadas; realizar os actos conservatórios dos direitos relativos aos bens comuns; regular o uso das coisas comuns e a prestação dos serviços de interesse comum; executar as deliberações da Assembleia; representar o conjunto dos condóminos perante as autoridades administrativas; prestar contas à Assembleia; assegurar a execução do regulamento e das disposições legais e administrativas relativas ao Condomínio e guardar e manter todos os documentos que digam respeito ao Condomínio. O administrador pode executar todas estas funções na sua aplicação.

1.2.2 O Síndico

Definição de síndico: Indivíduo eleito entre os membros de uma associação ou de uma classe para zelar e defender os interesses da mesma.

Cabe ao síndico exercer a administração interna do condomínio, referente à vigilância, moralidade e segurança. Seleccionar, admitir e demitir funcionários fixando-lhes os salários de acordo com a verba do orçamento do ano, respeitando o nível salarial da categoria. Convocar assembleia-geral e presta contas da contabilidade do condomínio. Escolher empresas prestadoras de serviços ou terceiros para execução de obras pré-estabelecidas e aprovadas em assembleia. Praticar os actos que lhe atribuírem a lei do condomínio, a convenção e o regimento interno.

1.2.3 O Condómino (Morador)

Definição de morador: habita, reside, ocupa de forma habitual e contínua uma determinada residência. Cabe ao morador, respeitar os direitos dos moradores, funcionários e vizinhos; pagar em dia os valores relativos à quota-parte das despesas ordinárias; pagar, sempre que exigida, a quota-parte nas despesas com obras necessárias à manutenção do prédio, mesmo que o objecto da despesa não seja de seu interesse; contribuir para a constituição do fundo de reserva; não alterar a forma externa da fachada; não usar a unidade para fins diferentes da utilização do condomínio, como o funcionamento de uma empresa; não impedir o uso das

⁶ Para evitar um condomínio sem gestão quando um mandato chega ao fim, o antigo administrador mantém-se em funções até ser eleito o sucessor.

partes comuns do condomínio; manter e conservar a área comum do condomínio, que é compartilhada com todos os condôminos, tal como *hall* de entrada, sala de festas, garagens e piscina; não decorar as partes externas do prédio com tonalidades ou cores diferentes das utilizadas no conjunto do bloco.

2. Método – Desenvolvimento do Protótipo

O presente trabalho consiste num protótipo funcional de SmartCondo, foi desenvolvido na ferramenta de desenvolvimento Visual Studio 2008, através da plataforma ASP.net e com a linguagem de programação C#.

2.1. Engenharia de Requisitos – Requisitos identificados antes do desenvolvimento do Sistema

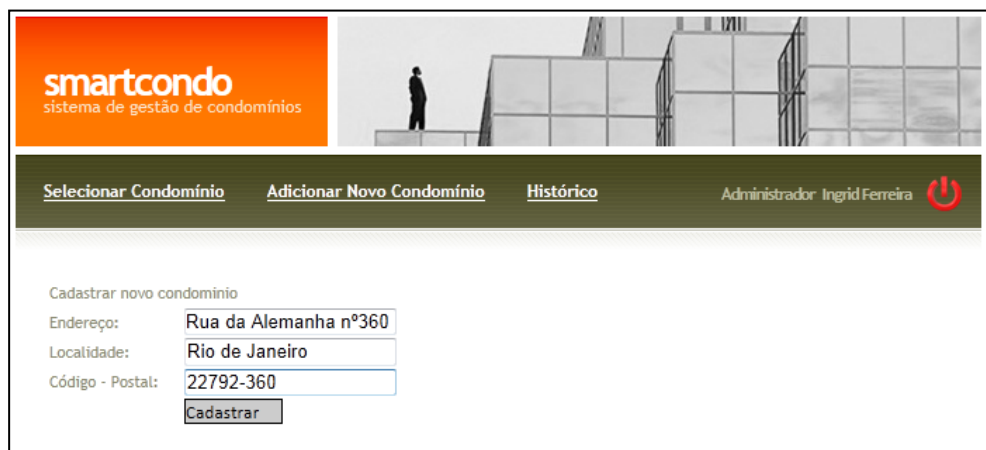
Após análise e levantamento das necessidades do sistema SmartCondo, foi criada uma tabela que tem como resultado os requisitos do sistema.

TABELA RESULTANTE DA ENGENHARIA DE REQUISITOS			
	Administrador	Síndico	Morador
Listar, inserir, editar e remover condomínio	X		
Listar, inserir, editar e remover morador	X		
Cadastrar área privada do condomínio	X		
Lançar despesas do condomínio	X		
Gerar e lançar cota mensal do condomínio	X		
Consultar cotas do condomínio lançadas	X		
Efectuar pagamento das despesas dos condomínios	X		
Exibir pagamentos realizados de despesas do condomínio	X		
Pagar cotas do condomínio		X	X
Consultar pagamentos realizados de cotas do condomínio	X	X	X
Reservar área privada do condomínio		X	X
Aprovar ou reprovar reservas solicitadas de área privada		X	
Convocar assembléia com aviso aos moradores via email		X	
Gerar relatórios customizados com informações do sistema		X	

2.2. Funcionalidades Implementadas no Sistema

2.2.1. Adicionar Novo Condomínio

O administrador preenche um formulário de inscrição onde insere o endereço, a localidade e o código postal do condomínio. Ao confirmar o cadastro, o condomínio fica com seu estado activo.



A imagem mostra a interface de usuário do sistema "smartcondo". No topo, há um cabeçalho com o logo "smartcondo" e o subtítulo "sistema de gestão de condomínios". Abaixo do cabeçalho, há uma barra de navegação com os links "Selecionar Condomínio", "Adicionar Novo Condomínio" (destacado), "Histórico" e o nome do usuário "Administrador Ingrid Ferreira" com um ícone de logout. O formulário principal, intitulado "Cadastrar novo condomínio", contém os seguintes campos: "Endereço:" com o valor "Rua da Alemanha nº360", "Localidade:" com o valor "Rio de Janeiro" e "Código - Postal:" com o valor "22792-360". Abaixo dos campos, há um botão "Cadastrar".


Figura 1 – Página Adicionar Novo Condomínio


2.2.2. Seleccionar Condomínio

O administrador visualiza uma lista com todos os condomínios activos. Essa lista contém a informação do número de identificação, o endereço, a localidade e o contrato do condomínio. A partir desta lista o administrador pode seleccionar um condomínio para realizar actividades específicas a este condomínio.



sistema de gestão de condomínios



[Selecionar Condomínio](#)
[Adicionar Novo Condomínio](#)
[Histórico](#)
Administrador Ingrid Ferreira 

Condomínios com contrato ativo:

ID	Condomínio	Localidade	Contrato	
4	Rua da Quitanda nº167	Rio de Janeiro	Ativo	Selecionar
6	Rua Antenor Frota nº235	Fortaleza	Ativo	Selecionar
7	Rua Bento Fernandes nº22	Rio de Janeiro	Ativo	Selecionar
8	Rua Américo Moraes nº1260	Vitória	Ativo	Selecionar
9	Rua Colaride nº5	Cacém	Ativo	Selecionar
10	Avenida Brasil	Rio de Janeiro	Ativo	Selecionar

[Gerar relatório com lista de moradores](#)

Figura 2 – Página Selecionar Condomínio

2.2.3. Histórico do Condomínio

O administrador visualiza uma lista com todos os condomínios activos e desactivos. Esta lista contém o número de identificação, o código postal e a localidade do condomínio, bem como a data de criação, o estado do contrato e a data de desactivação (se for o caso) do condomínio. Observações em relação à desactivação do condomínio do sistema podem ser visualizadas nesta lista.



sistema de gestão de condomínios



[Selecionar Condomínio](#)
[Adicionar Novo Condomínio](#)
[Histórico](#)
Administrador Ingrid Ferreira 

Lista de todos os condomínios:

ID	Condomínio	Código-Postal	Localidade	Criação	Contrato	Desativado em	Observações
10	Avenida Brasil	27775-390	Rio de Janeiro	20-09-2011	Ativo		
9	Rua Colaride nº5	22793-200	Cacém	20-09-2011	Ativo		
8	Rua Américo Moraes nº1260	24355-390	Vitória	18-09-2011	Ativo		
7	Rua Bento Fernandes nº22	22390-397	Rio de Janeiro	18-09-2011	Ativo		
6	Rua Antenor Frota nº235	60020-350	Fortaleza	18-09-2011	Ativo		
5	Rua Joaquim Nabuco nº360	22790-290	Fortaleza	18-09-2011	Desativo	19-09-2011	fim do contrato
4	Rua da Quitanda nº167	60020-360	Rio de Janeiro	18-09-2011	Ativo		

Figura 3 – Página Histórico do Condomínio

2.2.4. Editar Dados Condomínio


O administrador visualiza uma lista que contém a identificação, o endereço, o código postal e a localidade do condomínio, e consegue alterar, excepto a identificação, os dados da lista.




Figura 4 – Página Editar Dados Condomínio


2.2.5. Inserir Área Privada

O administrador visualiza uma lista com todas as áreas privadas já registradas no sistema. Esta lista contém a descrição da área privada, o tempo de reserva estimado a essa área e o valor da reserva. Além de visualizar as áreas já cadastradas, o administrador pode registrar novas áreas. Para isso deve preencher um formulário inserindo a descrição da área, o tempo estimado de reserva e o valor da reserva.



sistema de gestão de condomínios



[Selecionar Condomínio](#)
[Adicionar Novo Condomínio](#)
[Histórico](#)
Administrador Ingrid Ferreira


Selecionar Condomínio >> Rua da Quitanda nº167 >> Inserir Área Privada

[Editar Dados Condomínio](#)

[Inserir Área Privada](#)

[Moradores](#)

[Listar](#)

[Cadastrar Novo](#)

[Editar Dados](#)

[Remover](#)

[Gerir Contas](#)

[Lançar Despesa](#)

[Calcular Cota](#)

[Cotas Lançadas](#)

[Pagamentos](#)

[Histórico Pagamentos](#)

[Cancelar Contrato](#)

Áreas privadas do condomínio: Rua da Quitanda nº167

Área	Tempo Reserva	Valor
Sala de Reunião	2 horas	10,50
Sala de Jogos	3 horas	20,00
Sala de Festas	4 horas	100,00
Sala Infantil	2 horas	20,00
Sauna	1 hora	15,00
12		

Insira novas áreas privadas existentes no condomínio.

Tipo Área

Sala de Reunião

Tempo Estimado Reserva:

2 horas

Valor:

50,00

(99,99)

Inserir

Figura 5 – Página Inserir Área Privada

2.2.6. Cancelar Contracto do Condomínio

O administrador preenche um formulário com a informação do motivo do cancelamento do condomínio e confirma o cancelamento. Ao cancelar um condomínio, imediatamente todos os moradores ligados a este condomínio são desactivados.



sistema de gestão de condomínios



[Selecionar Condomínio](#)
[Adicionar Novo Condomínio](#)
[Histórico](#)
Administrador Ingrid Ferreira


Selecionar Condomínio >> Rua da Quitanda nº167 >> Cancelar Contrato

[Editar Dados Condomínio](#)

[Inserir Área Privada](#)

[Moradores](#)

[Listar](#)

[Cadastrar Novo](#)

Cancelar contrato do condomínio: Rua da Quitanda nº167

Motivo cancelamento:

Fim do contracto.

Cancelar



Figura 6 – Página Cancelar Contracto do Condomínio

2.2.7. Registar Novo Morador


O administrador preenche um formulário de inscrição onde insere o perfil (morador ou síndico), o nome, o apartamento, o contacto e o email do morador. Ao confirmar o registo, o morador fica com seu estado activo. Em seguida, o morador recebe em seu email os dados para autenticação no Sistema, tais como o *Utilizador* e *Código de acesso*. Para maior segurança no sistema, a senha do utilizador é encriptada através do algoritmo de encriptação MD5.

The screenshot shows the 'smartcondo' web application interface. The header includes the logo 'smartcondo sistema de gestão de condomínios' and a navigation bar with links: 'Selecionar Condomínio', 'Adicionar Novo Condomínio', 'Histórico', and 'Administrador Ingrid Ferreira' with a power icon. The main content area shows a breadcrumb trail: 'Selecionar Condomínio >> Rua da Quitanda nº167 >> Moradores >> Cadastrar Novo'. On the left, there are links for 'Editar Dados Condomínio', 'Inserir Área Privada', 'Moradores' (with sub-links 'Listar', 'Cadastrar Novo', 'Editar Dados', 'Remover'), and 'Gerir Contas'. The main form area is titled 'Cadastrar novo morador.' and shows the selected 'Condomínio: Rua da Quitanda nº167'. The form fields are: 'Perfil:' (dropdown menu set to 'Morador'), 'Nome:' (text box with 'Ingrid Ferreira'), 'Apto:' (text box with '101'), 'Contato:' (text box with '99999999'), and 'Email:' (text box with 'ingridmotte@hotmail.com'). A 'Cadastrar' button is at the bottom of the form. A back arrow is visible at the bottom left of the form area.


Figura 7 – Página Registar Novo Morador


2.2.8. Editar Dados Morador

O administrador visualiza uma lista com a identificação, o nome, o contacto, e o email do morador e consegue alterar, excepto a identificação e os dados da lista.



sistema de gestão de condomínios



[Selecionar Condomínio](#)
[Adicionar Novo Condomínio](#)
[Histórico](#)
Administrador Ingrid Ferreira


[Selecionar Condomínio](#)
>>
Rua da Quitanda nº167
>>
Moradores
>>
[Editar Dados](#)

[Editar Dados Condomínio](#)

[Inserir Área Privada](#)

[Moradores](#)

[Listar](#)

[Cadastrar Novo](#)

[Editar Dados](#)

[Remover](#)

[Gerir Contas](#)

[Lançar Despesa](#)

[Calcular Quota](#)

[Quotas Lançadas](#)

[Pagamentos](#)


Rua da Quitanda nº167

ID	Nome	Contato	Email	
4	Francisco Soares	999999999	fancis@condo.com	Alterar
5	Francisco Ferreira	999999999	jaferreira@coca-cola.com	Alterar
6	Maria Pereira	999999999	marp@condo.com	Alterar
7	João Silva	999999999	ingridmotte@hotmail.com	Alterar
8	José Sampaio	999999999	josesamp@condo.com	Alterar
9	Viviane Souza	999999999	vivisouza@condo.com	Alterar
10	Larissa Fraga	999999999	larissa@condo.com	Alterar
11	Oswaldo Motté	999999999	oswaldom@condo.com	Alterar
12	Bruno Farias	999999999	brunoF@condo.com	Alterar
13	Juliana Gaya	999999999	jujuga@condo.com	Alterar
12				


Figura 8 – Página Editar Dados Morador


2.2.9. Listar Moradores

O administrador visualiza uma lista com os moradores activos e desactivados. Esta lista contém o número de identificação, o nome, o apartamento, o contacto, o email e o estado do cadastro do morador.



sistema de gestão de condomínios



[Selecionar Condomínio](#)
[Adicionar Novo Condomínio](#)
[Histórico](#)
Administrador Ingrid Ferreira


Selecionar Condomínio >> Rua da Quitanda nº167 >> Moradores

[Editar Dados Condomínio](#)

[Inserir Área Privada](#)

[Moradores](#)

[Listar](#)

[Cadastrar Novo](#)

[Editar Dados](#)

[Remover](#)

[Gerir Contas](#)

[Lançar Despesa](#)

[Calcular Quota](#)

[Quotas Lançadas](#)

[Pagamentos](#)

[Histórico Pagamentos](#)

Condomínio: Rua da Quitanda nº167
Síndico responsável: João Silva - Apto 201

ID	Utilizador	Apto	Contato	Email	Entrada	Cadastro
4	Francisco Soares	101	999999999	fancis@condo.com	18-09-2011	Ativo
5	Francisco Ferreira	101	999999999	jaferreira@coca-cola.com	18-09-2011	Ativo
6	Maria Pereira	102	999999999	marp@condo.com	19-09-2011	Ativo
7	João Silva	201	999999999	ingridmotte@hotmail.com	19-09-2011	Ativo
8	José Sampaio	202	999999999	josesamp@condo.com	19-09-2011	Ativo
9	Viviane Souza	301	999999999	vivisouza@condo.com	19-09-2011	Ativo
10	Larissa Fraga	302	999999999	larissa@condo.com	19-09-2011	Ativo
11	Oswaldo Motté	401	999999999	oswaldom@condo.com	19-09-2011	Ativo
12	Bruno Farias	402	999999999	brunoF@condo.com	19-09-2011	Ativo
13	Juliana Gaya	501	999999999	jujuga@condo.com	19-09-2011	Ativo

Figura 9 – Página Listar Moradores

2.2.10. Remover Moradores

O administrador preenche um formulário informando os dados do morador a ser desactivado, o motivo do cancelamento e confirma o cancelamento. O estado do morador após o cancelamento é alterado para desactivado.



sistema de gestão de condomínios



[Selecionar Condomínio](#)
[Adicionar Novo Condomínio](#)
[Histórico](#)
Administrador Ingrid Ferreira


Selecionar Condomínio >> Rua da Quitanda nº167 >> Moradores >> Remover

[Editar Dados Condomínio](#)

[Inserir Área Privada](#)

[Moradores](#)

[Listar](#)

Condomínio: Rua da Quitanda nº167
Moradores ativos: Francisco Soares 
Motivo da remoção: Mudou-se do condomínio Remover

Figura 10 – Página Remover Moradores

2.2.11. Lançar Despesa Condomínio

O administrador preenche um formulário onde insere o tipo de conta (tipos de despesas que o condomínio pode ter), o nº da factura, o valor, a data de vencimento, o mês de referência e o ano de referência da despesa. Opcionalmente pode inserir informações extras, desde que relevantes. Os dados desta despesa lançada será utilizado posteriormente para realizar o cálculo da quota mensal do condomínio.



The screenshot displays the 'smartcondo' web application interface. At the top, there is a navigation bar with links: 'Selecionar Condomínio', 'Adicionar Novo Condomínio', and 'Histórico'. The user is logged in as 'Administrador Ingrid Ferreira'. The breadcrumb trail shows the path: 'Selecionar Condomínio >> Rua da Quitanda nº167 >> Gerir Contas >> Lançamentos'. The main heading is 'Efetuar lançamento de conta do condomínio Rua da Quitanda nº167.'. On the left, there is a sidebar menu with options: 'Editar Dados Condomínio', 'Inserir Área Privada', 'Moradores', 'Listar', 'Cadastrar Novo', 'Editar Dados', 'Remover', 'Gerir Contas', 'Lançar Despesa', and 'Calcular Quota'. The main form contains the following fields: 'Tipo de Conta:' (dropdown menu set to 'Água'), 'Fatura:' (text input '236022'), 'Valor:' (text input '300,00'), 'Vencimento:' (calendar icon and date '10/10/2011'), 'Mês Referente:' (dropdown menu set to 'Setembro'), 'Ano Referente:' (dropdown menu set to '2011'), and 'Observações:' (text input 'sem observações'). A 'Lançar' button is at the bottom of the form.

Figura 11 – Página Lançar Despesa Condomínio

2.2.12. Calcular Cota do Condomínio

O administrador visualiza o endereço do condomínio e o respectivo número de moradores. Ao continuar o processo, selecciona o mês e o ano da quota que deseja calcular e lançar. Se o mês e o ano já tiverem sido lançados anteriormente, o sistema não deixa prosseguir e informa que já existe nesse período uma cota lançada. Se a quota não tiver sido lançada anteriormente, o administrador consegue visualizar as despesas do condomínio realizadas no mês e ano solicitados. O administrador então selecciona o valor total dessas despesas e obtém o valor da quota mensal para cada morador do condomínio. O administrador insere o vencimento da quota, que é verificado pelo sistema se é coerente, e emite a quota no sistema. Ao emitir, a quota fica disponibilizada aos moradores deste condomínio para pagamento.

smartcondo
sistema de gestão de condomínios

Seleccionar Condomínio Adicionar Novo Condomínio Histórico Administrador Ingrid Ferreira

Seleccionar Condomínio >> Rua da Quitanda nº167 >> Gerir Contas >> Calcular Quota

[Editar Dados Condomínio](#)
[Inserir Área Privada](#)
[Moradores](#)
[Listar](#)
[Cadastrar Novo](#)
[Editar Dados](#)
[Remover](#)
[Gerir Contas](#)
[Lançar Despesa](#)
[Calcular Quota](#)
[Quotas Lançadas](#)
[Pagamentos](#)
[Histórico Pagamentos](#)

Condomínio	Quantidade Moradores	
Rua da Quitanda nº167	11	Continuar

Selecione o mês e o ano para calcular a despesa gerada pelo condomínio:

Mês: Ano:

[Visualizar despesa do condomínio](#)

Mes	Ano	Condomínio	Total Despesa	
Julho	2011	Rua da Quitanda nº167	500,00	Seleccionar

Valor para cada morador: 55,45
Vencimento:

[Emitir Quota](#)

Figura 12 – Página Calcular Quota do Condomínio

2.2.13. Visualizar Quotas Lançadas

O administrador visualiza todas as quotas que foram lançadas por ele no Sistema. É informado o código, o mês e o ano referente, o vencimento e o valor da quota.

smartcondo
sistema de gestão de condomínios

Seleccionar Condomínio Adicionar Novo Condomínio Histórico Administrador Ingrid Ferreira

Seleccionar Condomínio >> Rua da Quitanda nº167 >> Gerir Contas >> Quotas Lançadas

[Editar Dados Condomínio](#)
[Inserir Área Privada](#)
[Moradores](#)
[Listar](#)
[Cadastrar Novo](#)
[Editar Dados](#)

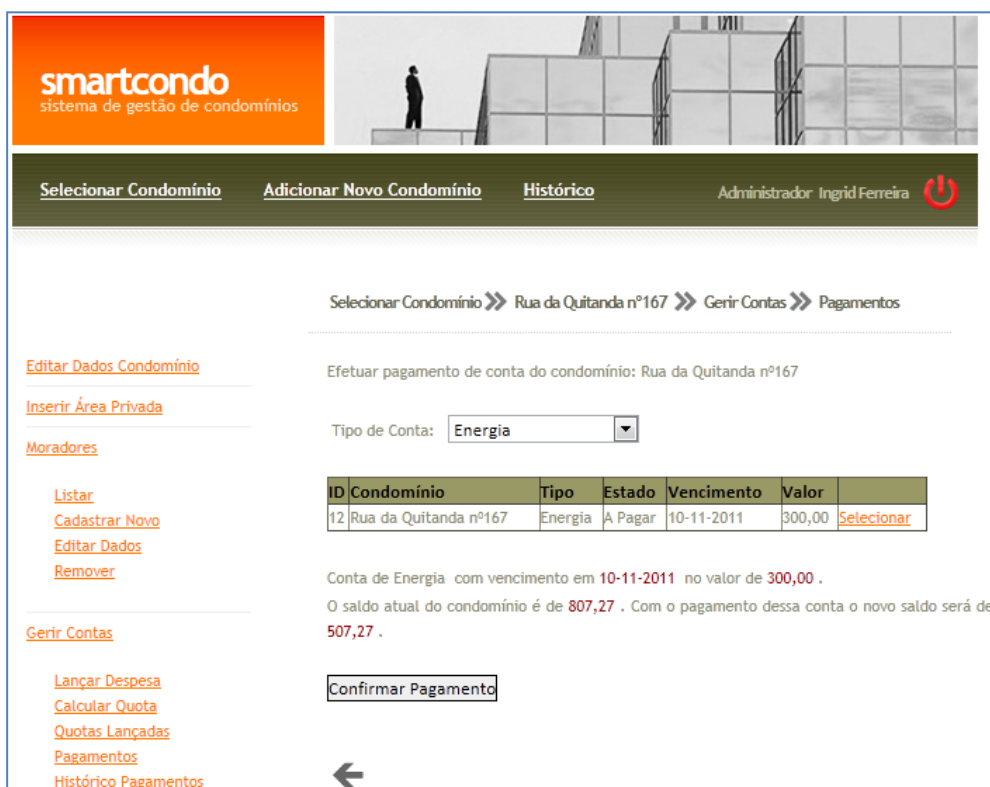
Lista de quotas lançadas do condomínio: Rua da Quitanda nº167 .

Código	Mês	Ano	Vencimento	Valor
5	Setembro	2011	10-10-2011	70,00
6	Outubro	2011	10-11-2011	37,27
7	Novembro	2011	10-12-2011	60,91

Figura 13 – Página Visualizar Quotas Lançadas

2.2.14. Pagamentos Contas do Condomínio

O administrador visualiza todas as contas do condomínio que estão pendente para pagamento. Seleciona o tipo de conta que deseja pagar e é lhe exibido o saldo que o condomínio possui e o saldo resultante do condomínio após o pagamento. O administrador confirma o pagamento, o estado da conta é alterado de *A Pagar* para *Paga* e por fim, é debitado do fundo do condomínio (saldo que o condomínio possui) o valor da conta paga.



smartcondo
sistema de gestão de condomínios

Selecionar Condomínio Adicionar Novo Condomínio Histórico Administrador Ingrid Ferreira

Selecionar Condomínio >> Rua da Quitanda nº167 >> Gerir Contas >> Pagamentos

Editar Dados Condomínio
Inserir Área Privada
Moradores
Listar
Cadastrar Novo
Editar Dados
Remover
Gerir Contas
Lançar Despesa
Calcular Quota
Quotas Lançadas
Pagamentos
Histórico Pagamentos

Efetuar pagamento de conta do condomínio: Rua da Quitanda nº167

Tipo de Conta: Energia

ID Condomínio	Tipo	Estado	Vencimento	Valor	
12	Rua da Quitanda nº167	Energia	A Pagar	10-11-2011	300,00

Conta de Energia com vencimento em 10-11-2011 no valor de 300,00 .
O saldo atual do condomínio é de 807,27 . Com o pagamento dessa conta o novo saldo será de 507,27 .

Confirmar Pagamento

Figura 14 – Página Pagamentos Contas do Condomínio

2.2.15. Histórico do Pagamento de Contas do Condomínio

O administrador visualiza as contas que foram pagas. É informado o código da conta, o endereço do condomínio, o tipo de conta, o estado (Paga) e o vencimento, o valor e a data em que o pagamento foi realizado.

smartcondo
sistema de gestão de condomínios

Selecionar Condomínio Adicionar Novo Condomínio Histórico Administrador Ingrid Ferreira

Selecionar Condomínio >> Rua da Quitanda nº167 >> Gerir Contas >> Histórico

[Editar Dados Condomínio](#)
[Inserir Área Privada](#)
[Moradores](#)
[Listar](#)
[Cadastrar Novo](#)
[Editar Dados](#)
[Remover](#)

Lista de contas pagas do condomínio: Rua da Quitanda nº167

ID	Condomínio	Tipo	Estado	Vencimento	Valor	Pago em
1	Rua da Quitanda nº167	Água	Paga	05-10-2011	300,00	20-09-2011
2	Rua da Quitanda nº167	Energia	Paga	05-10-2011	200,00	20-09-2011
3	Rua da Quitanda nº167	Limpeza	Paga	05-10-2011	100,00	20-09-2011
13	Rua da Quitanda nº167	Manutenção técnica	Paga	10-12-2011	560,00	22-09-2011

Figura 15 – Página Histórico do Pagamento de Contas do Condomínio

2.2.16. Pagamentos Morador

O morador visualiza as quotas mensais que estão pendentes de pagamento. É informado o código da quota, o nome e o apartamento do morador, o mês, ano, valor, vencimento e estado (*A Pagar*) referente à quota. O morador selecciona a quota para efectuar o pagamento, são exibidas as informações para a realização do pagamento, tais como entidade, referência multibanco e valor da quota. O número de entidade é um valor fixo para cada condomínio e a referência multibanco é gerada através de uma função *Random* que tem como resultados a geração de números aleatórios. O morador ao confirmar o pagamento através do botão pagar, altera o estado da quota de *A Pagar* para *Paga* e o valor do fundo do condomínio é acrescido com o valor da quota paga. Se houver quotas atrasadas, é informado ao morador através de um aviso que, além de informar a dívida, solicita a regularização da mesma.



sistema de gestão de condomínios



Morador Maria Pereira


Área Morador
Morador >> Maria Pereira >> Rua da Quitanda nº167 >> Pagamentos

[Perfil](#)

[Histórico](#)

[Pagamentos](#)

[Reserva de Espaço](#)

[Agenda](#)

Lista de quotas para pagamento.

Quota	Morador	Apto	Mês	Ano	Valor	Vencimento	Estado	
7	Maria Pereira	102	Novembro	2011	60,91	10-12-2011	A Pagar	Selecionar

Entidade: 10404
Referência: 975688293
Valor: 60,91

Pagar

Figura 16 – Página Pagamentos Morador

2.2.17. Histórico Pagamento Morador

O morador visualiza uma lista com a informação das quotas mensais pagas por ele. Esta lista contém a informação do código da quota, o nome e o apartamento do morador, o mês, ano, valor e o estado da quota (*Paga*).



sistema de gestão de condomínios



Morador Maria Pereira


Área Morador
Morador >> Maria Pereira >> Rua da Quitanda nº167 >> Histórico

[Perfil](#)

[Histórico](#)

[Pagamentos](#)

[Reserva de Espaço](#)

[Agenda](#)

Lista de todas as quotas pagas.

Quota	Morador	Apto	Mês	Ano	Valor	Estado
5	Maria Pereira	102	Setembro	2011	70,00	Paga
6	Maria Pereira	102	Outubro	2011	37,27	Paga

Figura 17 – Página Histórico Pagamento Morador

2.2.18. Reserva de Espaço

O morador visualiza uma lista com as informações das áreas privadas registadas condomínio. Esta lista possui a informação do código e a descrição da área privada, o tempo estimado de reserva e o preço para reserva. Após seleccionar a área desejada, o morador informa a data que deseja utilizar a área e submete o pedido de reserva. O morador fica a espera da aprovação ou reprovação do pedido que é analisado pelo síndico do condomínio.

The screenshot shows the 'smartcondo' system interface. At the top, there's a header with the logo and a navigation bar showing the user 'Morador Maria Pereira'. Below this, a breadcrumb trail reads: 'Morador >> Maria Pereira >> Rua da Quitanda nº167 >> Reserva de Espaço'. On the left, a sidebar menu lists options: 'Principal', 'Histórico', 'Pagamentos', 'Agenda', and 'Reserva de Espaço' (which is highlighted). The main content area is titled 'Condomínio: Rua da Quitanda nº167' and contains a table of available private areas for reservation. Below this table, there's a summary of the selected reservation and a date picker.

Código	Área	Tempo	Preço	
1	Sala de Reuniao	2 horas	10,50	Seleccionar
2	Sala de Jogos	3 horas	20,00	Seleccionar
3	Sala de Festas	4 horas	100,00	Seleccionar
4	Sala Infantil	2 horas	20,00	Seleccionar
5	Sauna	1 hora	15,00	Seleccionar
6	Academia	1 hora	15,00	Seleccionar
7	Piscina	4 horas	100,00	Seleccionar

Área	Tempo Utilização	Preço
Sala de Festas	4 horas	100,00

Data: 06/10/2011

[Solicitar Reserva](#)

Figura 18 – Página Reserva de Espaço

2.2.19. Gestão de Reserva

O síndico visualiza uma lista com as informações das reservas de áreas privadas que estão pendentes para aprovação. Esta lista contém a informação do código da reserva, o nome o apartamento do morador, a descrição da área, o tempo estimado de reserva (tempo de utilização da área reservada), o valor da reserva, o dia desejado para a reserva, o dia em que a reserva foi solicitada e o seu estado (*Aguarda Aprovação*). O síndico pode aprovar ou reprovar uma ou mais reservas ao mesmo tempo. Ao aprovar ou reprovar o pedido, a reserva tem o seu estado alterado.



sistema de gestão de condomínios



Sindico João Silva


Área Pessoal

Administrativo >> João Silva >> Rua da Quitanda nº167 >> Gestão de Reservas

Principal

Histórico

Pagamentos

Agenda

Reserva de Espaço

Condomínio: Rua da Quitanda nº167


ID	Morador	Apto	Área	Tempo	Valor	Reservar	Emitido	Estado
4	João Silva	201	Academia	1 hora	15,00	25-09-2011	21-09-2011	Aguarda Aprovação
6	João Silva	201	Sala de Reuniao	2 horas	10,50	28-10-2011	22-09-2011	Aguarda Aprovação
7	João Silva	201	Sauna	1 hora	15,00	30-09-2011	22-09-2011	Aguarda Aprovação
8	João Silva	201	Academia	1 hora	15,00	28-09-2011	22-09-2011	Aguarda Aprovação

Aprovar
Reprovar


Figura 19 – Página Gestão de Reserva


2.2.20. Agenda Morador

O morador visualiza uma lista com as informações das suas reservas de áreas privadas. Esta lista permite o acompanhamento do pedido de reserva realizado. Esta lista contém o nome e o apartamento do morador, a descrição da área, o dia em que a reserva foi solicitada, a duração da utilização da área, o valor da reserva e o estado o seu estado.



sistema de gestão de condomínios



Sindico João Silva


Área Pessoal

Pessoal >> João Silva >> Rua da Quitanda nº167 >> Agenda

Principal

Histórico

Pagamentos

Agenda

Reserva de Espaço

Reservas solicitadas:

Morador	Apto	Área	Dia Solicitado	Duração	Valor	Estado
João Silva	201	Academia	25-09-2011	1 hora	15,00	Aguarda Aprovação
João Silva	201	Academia	23-09-2011	1 hora	15,00	Reserva Aprovada
João Silva	201	Sala de Jogos	08-09-2011	3 horas	20,00	Reserva Aprovada
João Silva	201	Sala de Reuniao	02-09-2011	2 horas	10,50	Reserva Reprovada

Área Administrativa

Marcar Reunião

Gestão de Reservas

Relatório Mensal

Figura 20 – Página Agenda Morador

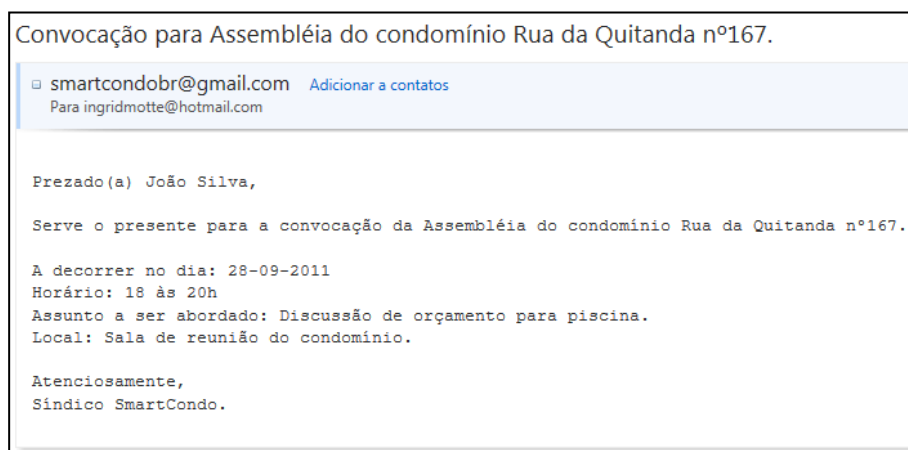
2.2.21. Convocar Assembleia

O síndico preenche um formulário para convocação da assembleia do condomínio. No formulário é informado a data da assembleia, o horário previsto, o assunto que será discutido e o local onde será realizada. Ao convocar, o síndico envia, em um único clique, email para todos os moradores correspondentes ao condomínio.



The screenshot shows the 'smartcondo' web application interface. The header includes the logo and a navigation bar with the user's name 'Síndico João Silva' and a power icon. The main content area is titled 'Convocação Assembleia' and contains a form to schedule a meeting. The form fields are: Data (28/09/2011), Horário (18 às 20h), Assunto (Discussão de orçamento), and Local (Sala de reunião do condomínio). A 'Convocar Assembleia' button is at the bottom of the form. On the left, there is a sidebar with links for 'Perfil', 'Histórico', 'Pagamentos', 'Reserva de Espaço', and 'Agenda'.

Figura 21 – Página Convocar Assembleia



The screenshot shows an email invitation for a condominium assembly. The subject is 'Convocação para Assembléia do condomínio Rua da Quitanda nº167.' The email is addressed to 'João Silva' and is from 'smartcondobr@gmail.com'. The body of the email contains the following information: 'Serve o presente para a convocação da Assembléia do condomínio Rua da Quitanda nº167.', 'A decorrer no dia: 28-09-2011', 'Horário: 18 às 20h', 'Assunto a ser abordado: Discussão de orçamento para piscina.', 'Local: Sala de reunião do condomínio.', and 'Atenciosamente, Síndico SmartCondo.'

Figura 22 – Email recebido pelo morador com a convocação da assembleia

2.2.22. Gerar Relatório com lista de moradores

O administrador visualiza relatório que possui os condomínios com seus respectivos moradores. Além de visualizar, o administrador pode exportar o relatório para um ficheiro em formato *pdf* ou Excel.

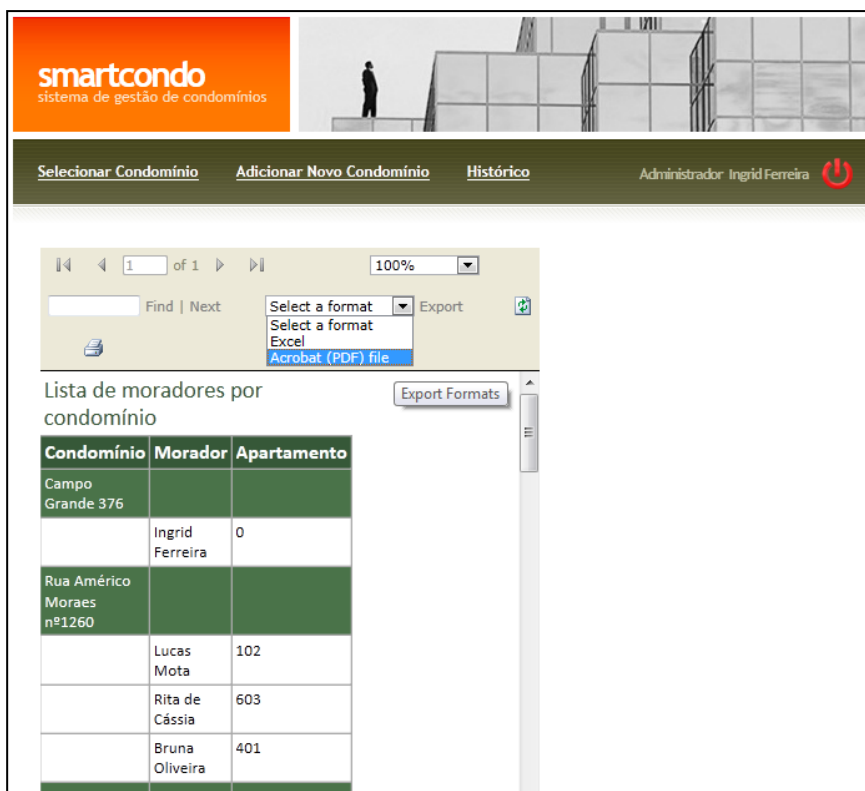


Figura 23 – Página Gerar relatório com lista de moradores

Lista de moradores por condomínio		
Condomínio	Morador	Apartamento
Campo Grande 376		
	Ingrid Ferreira	0
Rua Américo Moraes nº1260		
	Lucas Mota	102
	Rita de Cássia	603
	Bruna Oliveira	401
Rua Antenor Frota nº235		
	Ricky Momad	102
	Sabrina Souto	103
	Suzana Vieira	201
	Fausto Silva	202
	Luciano Huck	203
	Karine Carvalho	301
	Fábio Assunção	302
	Cristiano Ronaldo	303
	Isadora Motté	401
	Joaquim Rocha	402

Figura 24 – Relatório gerado em formato pdf

2.2.23. Autenticação

Na autenticação, os dados do utilizador são verificados na base de dados do sistema através da inserção de dados *Utilizador* e *Código de Acesso* informados pelo utilizador. Os campos são de preenchimento obrigatório. Só após a validação dos dados, o utilizador acede o sistema e a área correspondente ao seu perfil.

smartcondo
sistema de gestão de condomínios

A nossa empresa Os nossos serviços Onde estamos quinta-feira, 22 Setembro

Bem-vindo ao SmartCondo, a nossa empresa é especializada na gestão de condomínios residenciais. A SmartCondo é uma empresa situada em Lisboa que fornece soluções de software à indústria imobiliária desde Janeiro de 2011, desenvolvendo sistemas próprios especificamente projetados para este mercado.

Utilizador
sind

Código de acesso
●●●

Entrar

[Recuperar senha](#)

Figura 25 – Página Autenticação

2.2.24. Recuperação de Senha

A funcionalidade recuperar *password* permite ao utilizador que se esqueceu da sua senha ter acesso a uma nova senha. Para isso o utilizador insere o seu email num formulário. Se o email corresponder a um email anteriormente registado na base de dados do sistema, é gerada uma nova senha através de uma função *Random* e os novos dados de autenticação do utilizador são enviados por email.

Figura 26 – Página Recuperação de Senha

Figura 27 – Email de recuperação de senha

2.2.25. Logout

O *logout*, limpa todas as variáveis de sessão do sistema e envia o utilizador para a página inicial do sistema onde para ter acesso ao sistema terá que realizar nova autenticação.

2.3. Modelo de Dados

2.3.1 Representação UML – Use Case

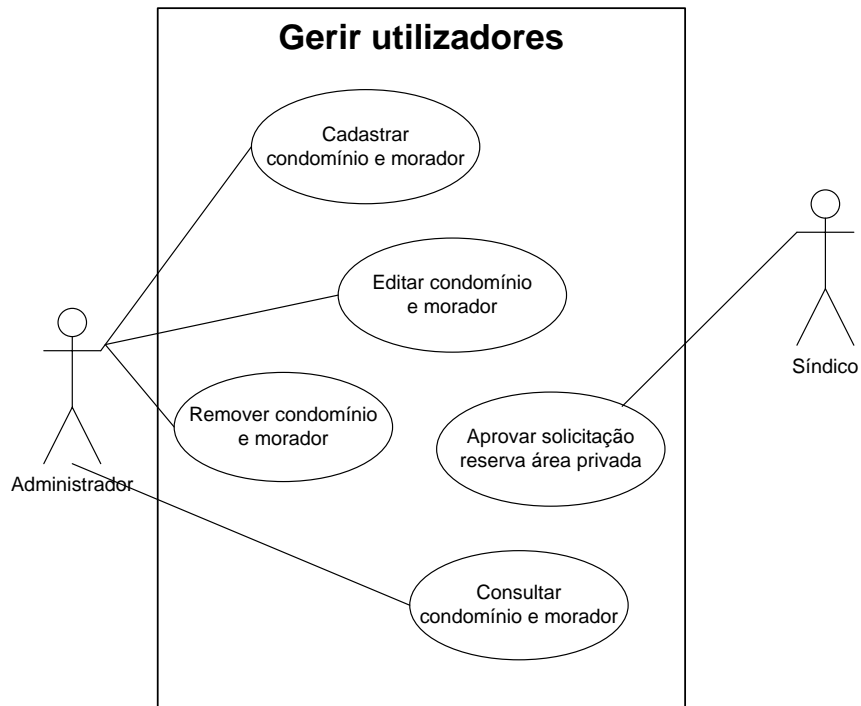


Figura 28 – UML Gerir Utilizadores

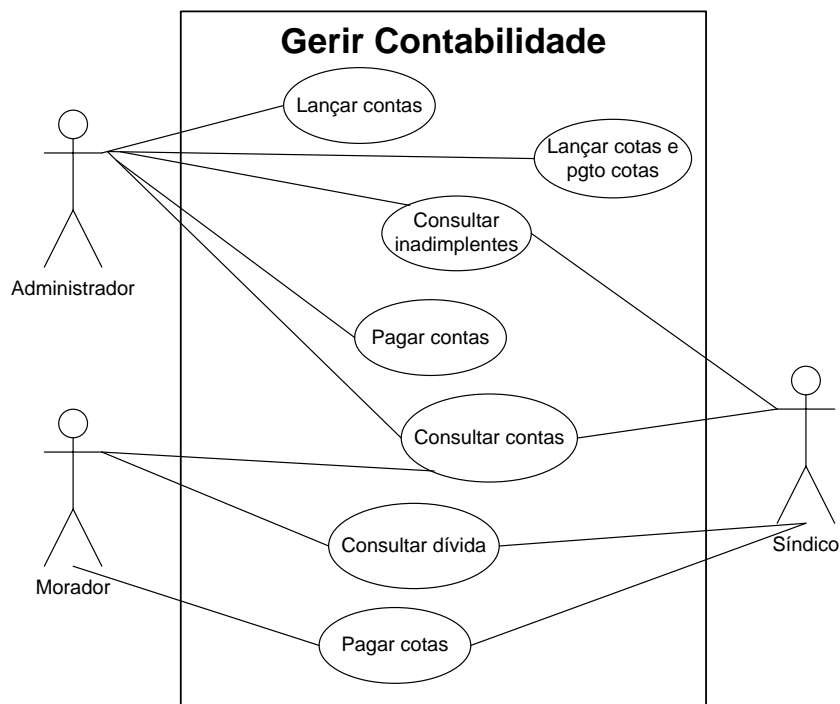


Figura 29 – UML Gerir Contabilidade

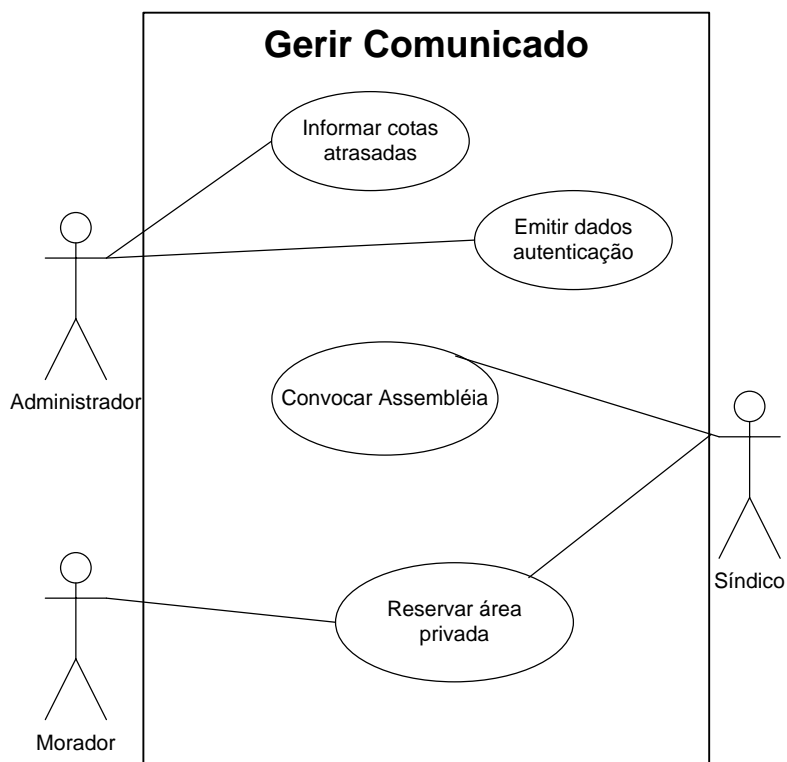


Figura 30 – UML Gerir Comunicado

2.3.2. Representação UML – Diagrama de Classes

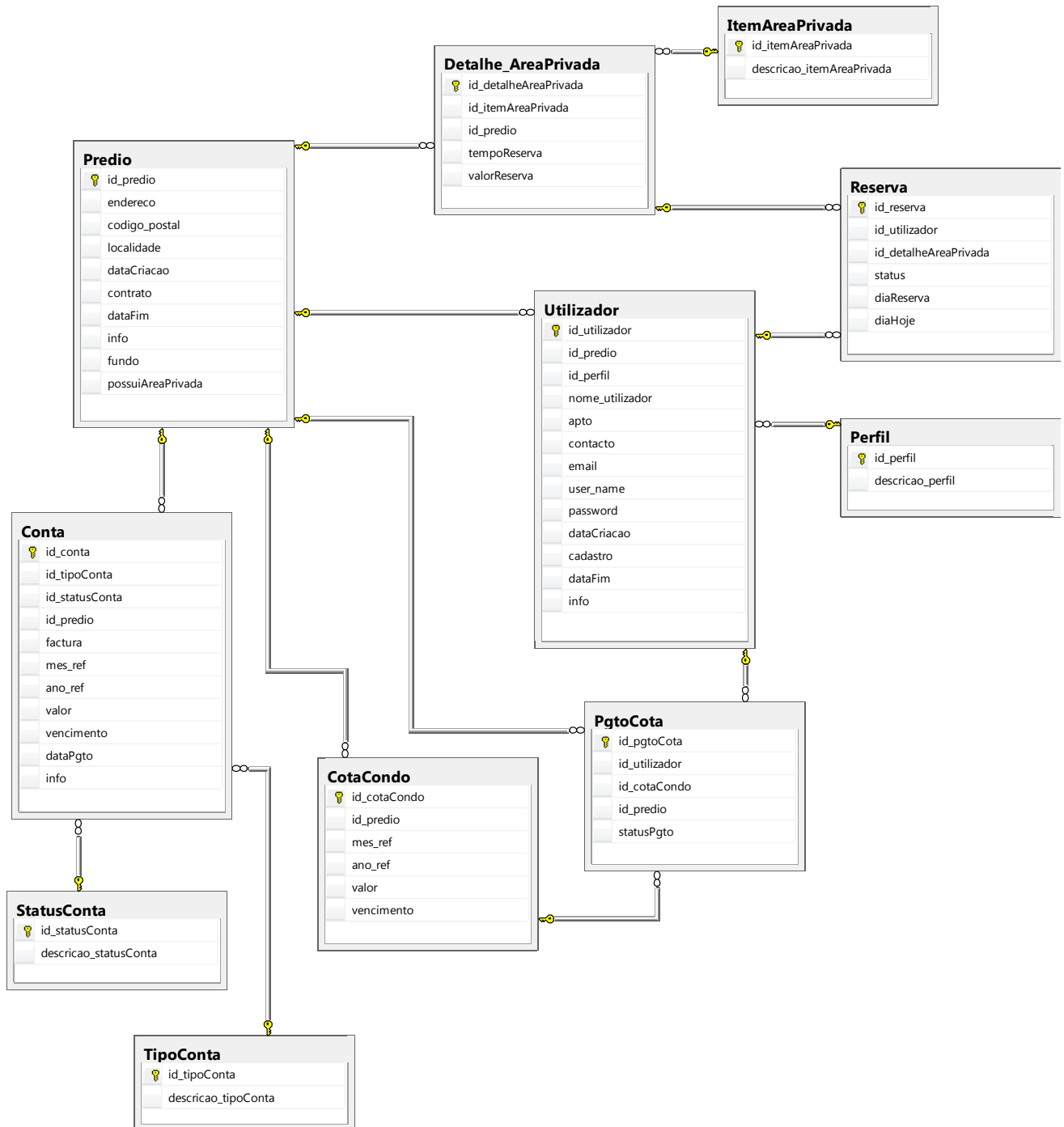


Figura 31 – Diagrama de Classes

2.3.3. Descrição das Tabelas

2.3.3.1. Tabela Conta

Nesta tabela são guardados as informações das contas do condomínio. É o administrador que envia estes dados à base de dados. Cada linha desta tabela é uma factura de uma conta de despesa de um determinado condomínio.

Tabela Conta				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_conta	chave primária da tabela	int	Não aceita	_____
id_tipoConta	chave estrangeira	int	Não aceita	Tipo_Conta
id_statusConta	chave estrangeira	int	Não aceita	Status_Conta
id_predio	chave estrangeira	int	Não aceita	Predio
factura	nº de identificação da factura descrito na factura em papel (factura real)	int	Não aceita	_____
mes_ref	mês referente à despesa, descrito na fatura em pape (factura real)	nvarchar(20)	Não aceita	_____
ano_ref	ano referente à despesa, descrito na facutra em papel (factura real)	nvarchar(20)	Não aceita	_____
valor	valor da despesa, descrito na factura em papel (factura real)	decimal(10,2)	Não aceita	_____
vencimento	data de pagamento da despesa, descrito na factura em papel (factura real)	datetime	Não aceita	_____
dataPgto	data em que a factura foi paga pelo administrador, valor retirado da data do sistema operativo no acto do pagamento	datetime	Aceita	_____
info	informação adicional optativa	nvarchar(MAX)	Aceita	_____

2.3.3.2. Tabela CotaCondo

Nesta tabela são guardados as informações das quotas dos condomínios. É o administrador que envia esses dados à base de dados. Cada linha dessa tabela é uma quota de um determinado mês e ano de um condomínio.

Tabela CotaCondo				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_cotaCondo	chave primária da tabela	int	Não aceita	_____
id_predio	chave estrangeira	int	Não aceita	Predio
mes_ref	mês referente à cota do condomínio	nvarchar(20)	Não aceita	_____
ano_ref	ano referente à cota do condomínio	nvarchar(20)	Não aceita	_____
valor	valor da cota	decimal(10,2)	Não aceita	_____
vencimento	data de pagamento da cota	datetime	Não aceita	_____

2.3.3.3. Tabela Detalhe_AreaPrivada

Nesta tabela são guardados as informações das áreas privadas existentes em cada condomínio. É o administrador que envia esses dados à base de dados. Cada linha dessa tabela é o registo de uma área privada em um condomínio com informações para reservas. É nessa tabela que os moradores buscam informações para realizar a reserva do espaço privado.

Tabela Detalhe_AreaPrivada				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_detalheAreaPrivada	chave primária da tabela	int	Não aceita	_____
id_itemAreaPrivada	chave estrangeira	int	Não aceita	ItemAreaPrivada
id_predio	chave estrangeira	int	Não aceita	Predio
tempoReserva	tempo de reserva de cada área privada	nvarchar(50)	Não aceita	_____
valorReserva	valor da reserva de cada área privada	decimal(10,2)	Não aceita	_____

2.3.3.4. Tabela ItemAreaPrivada

Nesta tabela são guardados a descrição das áreas privadas existentes em cada condomínio. Esses dados já são inseridos pelo programador previamente na base de dados que são: Sala de Reunião, Sala de Jogos, Sala de Festas, Piscina, Academia, Quadra de Futebol, Quadra de Tênis, Sala Infantil e Sauna. Cada linha dessa tabela corresponde a um tipo de área privada que um condomínio pode possuir. O objectivo dessa tabela é passar a descrição das áreas privadas para a tabela Detalhe_AreaPrivada.

Tabela ItemAreaPrivada				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_itemAreaPrivada	chave primária da tabela	int	Não aceita	_____
descricao_itemAreaPrivada	descrição da área privada	nvarchar(50)	Não aceita	_____

2.3.3.5 .Tabela Perfil

Nesta tabela são guardados a descrição dos perfis que cada utilizador do sistema pode assumir. Esses dados já são inseridos pelo programador previamente na base de dados que são: morador e síndico. Cada linha dessa tabela corresponde a um tipo de utilizador que um condomínio pode possuir. O objectivo dessa tabela é passar o perfil dos utilizadores para a tabela Utilizador.

Tabela Perfil				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_perfil	chave primária da tabela	int	Não aceita	_____
descricao_perfil	descrição do perfil	nvarchar(50)	Não aceita	_____

2.3.3.6. Tabela PgtoCota

Nesta tabela são guardados as quotas que cada morador deve pagar. É o administrador que envia esses dados à base de dados ao calcular o valor de cada quota e lançar a mesma no sistema. Cada linha dessa tabela corresponde uma quota de um morador pertencente a um condomínio. O objectivo dessa tabela é disponibilizar as quotas de cada morador para que esse morador possa posteriormente pagá-la.

Tabela PgtoCota				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_pgtoCota	chave primária da tabela	int	Não aceita	_____
id_utilizador	chave estrangeira	int	Não aceita	Utilizador
id_cotaCondo	chave estrangeira	int	Não aceita	CotaCondo
id_predio	chave estrangeira	int	Não aceita	Predio
statusPgto	valor da reserva de cada área privada	nvarchar(50)	Não aceita	_____

2.3.3.7. Tabela Predio

Nesta tabela são guardados as informações referentes a cada condomínio. É o administrador que envia esses dados à base de dados ao registar o condomínio no sistema. Cada linha dessa tabela corresponde a um condomínio. O objectivo dessa tabela é disponibilizar a informação de cada condomínio para a realização das funcionalidades do sistema.

Tabela Predio				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_predio	chave primária da tabela	int	Não aceita	_____
endereco	endereço do condomínio	nvarchar(100)	Não aceita	_____
codigo_postal	código-postal do condomínio	nvarchar(50)	Não aceita	_____
localidade	localidade do condomínio	nvarchar(50)	Não aceita	_____
dataCriacao	data do cadastro do condomínio	datetime	Não aceita	_____
contrato	estado do contracto do condomínio: activo (contracto vigente) ou desactivo (contracto cancelado)	nvarchar(50)	Não aceita	_____
dataFim	data do encerramento do contracto do condomínio, caso isso aconteça	datetime	Aceita	_____
info	informação adicional sobre o motivo do cancelamento do contracto do condomínio	nvarchar(MAX)	Aceita	_____
fundo	saldo disponível do condomínio	decimal(10,2)	Não aceita	_____
possuiAreaPrivada	informação se possui ou não alguma área privada no condomínio	nvarchar(20)	Aceita	_____

2.3.3.8. Tabela Reserva

Nesta tabela são guardados as informações referentes a reserva da área privada realizada por cada morador. É o morador que lança esses dados na base de dados ao solicitar a reserva no sistema. Cada linha dessa tabela corresponde a reserva solicitada por um morador de uma área privada em um determinado condomínio. O objectivo dessa tabela é além de registrar a reserva é dar informações ao síndico para posterior aprovação da mesma.

Tabela Reserva				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_reserva	chave primária da tabela	int	Não aceita	_____
id_utilizador	chave estrangeira	int	Não aceita	Utilizador
id_detalheAreaPrivada	chave estrangeira	int	Não aceita	Detalhe_AreaPrivada
status	estado da reserva: "Em espera", "Aprovada", "Reprovada"	nvarchar(20)	Não aceita	_____
diaReserva	dia pretendido para a reserva da área privada	datetime	Não aceita	_____
diaHoje	dia em que a reserva foi solicitada	datetime	Não aceita	_____

2.3.3.9. Tabela StatusConta

Nesta tabela são guardados a descrição dos possíveis estados que uma conta referente a uma despesa de um determinado condomínio possa assumir. Esses dados já são inseridos pelo programador previamente na base de dados que são: A Pagar, Pago, Atrasada. Cada linha dessa tabela corresponde a um estado que essa conta pode assumir. O objectivo dessa tabela é passar o estado da conta para a tabela Conta.

Tabela StatusConta				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_statusConta	chave primária da tabela	int	Não aceita	_____
descricao_statusConta	descrição do estado da conta referente a uma despesa do condomínio	nvarchar(50)	Não aceita	_____

2.3.3.10. Tabela TipoConta

Nesta tabela são guardados a descrição das possíveis despesas que um determinado condomínio pode assumir. Esses dados já são inseridos pelo programador previamente na base de dados que são: Água, Energia, Manutenção técnica, Limpeza e Funcionário. Cada linha dessa tabela corresponde a descrição de um tipo de despesa. O objectivo dessa tabela é passar o tipo de despesa do condomínio para a tabela Conta.

Tabela TipoConta				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_tipoConta	chave primária da tabela	int	Não aceita	_____
descricao_tipoConta	descrição dos tipos de despesas que um condomínio pode assumir	nvarchar(50)	Não aceita	_____

2.3.3.11. Tabela Utilizador

Nesta tabela são guardados as informações referentes a cada utilizador. É o administrador que lança esses dados na base de dados ao registar o morador no sistema, que além de morador pode assumir também a condição de síndico. Cada linha dessa tabela corresponde a um morador. O objectivo dessa tabela é disponibilizar a informação de cada morador para a realização das funcionalidades do sistema.

Tabela Utilizador				
Coluna	Descrição	Tipo de dados	Valor Nulo	Tab. Relacionamento
id_utilizador	chave primária da tabela	int	Não aceita	_____
id_predio	chave estrangeira	int	Não aceita	Predio
id_perfil	chave estrangeira	int	Não aceita	Perfil
nome_utilizador	nome do morador	nvarchar(50)	Não aceita	_____
apto	apto pertecente ao morador	nvarchar(20)	Não aceita	_____
contacto	contacto de telefone do morador	int	Não aceita	_____
email	email do morador	nvarchar(50)	Não aceita	_____
user_name	informação para autenticação no sistema	nvarchar(20)	Não aceita	_____
password	informação para autenticação no sistema	nvarchar(20)	Não aceita	_____
dataCriacao	data do cadastro do morador	datetime	Não aceita	_____
cadastro	estado do cadastro do morador: "Activo" ou "Desactivo"	nvarchar(20)	Não aceita	_____
dataFim	data da retirada do morador de um determinado condomínio, caso isso aconteça	datetime	Aceita	_____
info	informação adicional sobre o motivo da saída do morador do condomínio	nvarchar(MAX)	Aceita	_____

Conclusão

O presente trabalho final de curso foi particularmente útil para o meu percurso como estudante porque, com este trabalho pude implementar na prática várias metodologias e conhecimentos teóricos, adquiridos principalmente nas disciplinas de Análise e Concepção de Sistemas, Engenharia de Software e Interação Homem-Máquina.

Por outro lado, o recurso à linguagem de programação C#, permitiu consolidar os meus conhecimentos nesta linguagem de programação e consequentemente sinto mais a vontade para trabalhos futuros em C#.

Por outro lado, permitiu que ao longo do desenvolvimento deste trabalho fossem identificados problemas e encontradas soluções. Um dos principais problemas com que me deparei foi ter que colocar-me na visão do utilizador, e ao nível de interação homem-máquina utilizar soluções que facilitem a navegação do utilizador na plataforma, ou seja, que a interação seja intuitiva.

Com a elaboração deste protótipo adquiri também um conhecimento sobre a real funcionalidade e mais valia de implementar na prática de ferramentas de UML e modelação de dados. Quando iniciei este protótipo não recorri a estas ferramentas, o que implicou que a um determinado momento tivesse que reiniciar o projecto, e implementar de inicio a utilização de ferramentas UML e modelação de dados, de forma a que fossem pensadas as várias soluções possíveis antes de as colocar em prática e começar a escrever código.

Bibliografia

- Davis, M. (1990), *City of Quartz: Excavating the Future in Los Angeles*, Londres: Verso.
- Knox, P. (1992), The packaged landscapes of post-suburban America, in Whitehand, J. & Larkham, P. (eds.), *Urban Landscapes: International Perspectives*, (pp. 207-226). Londres: Routledge.
- Raposo, R. (2002), *Novas Paisagens: A Produção Social de Condomínios Fechados na Área Metropolitana de Lisboa*, Lisboa, ISEG/UTL (tese de doutoramento).
- Raposo, R. (2003). New landscapes: gated housing estates in the Lisbon Metropolitan Area, in *Geographica Helvetica*, 58 (4), pp. 293-301.
- Raposo, R. (2008). Condomínios fechados em Lisboa:paradigma e paisagem. *Análise Social* (vol. XLIII. 1º. Pp. 109-131).

Links:

<http://www.agendafacilsindico.com.br/manual.htm>

<http://www.infopedia.pt/lingua-portuguesa/>

<http://www.procon.df.gov.br/>

<http://www.licitamais.com.br/>

Anexos

Source Code

Código para autenticação no Sistema

```
namespace SmartCondo
{
    public partial class Site1 : System.Web.UI.MasterPage
    {
        static string strCon =
ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            DateTime agora = DateTime.Now;
            String hoje = String.Format("{0:dddd, d MMMM}", agora);
            Session.Add("data", hoje);
            lblHora.Text = Session["data"].ToString();
        }

        protected void LinkBtnRecuperaPass_Click(object sender, EventArgs e)
        {
            Response.Redirect("/Master1/RecuperaSenha.aspx", false);
        }

        protected void Button_Login_Click1(object sender, EventArgs e)
        {
            string query = "SELECT * FROM View_Utilizador_Predio WHERE user_name = @user_name
and password = @password";
            SqlDataReader dr;
            SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
            SqlCommand cmd = new SqlCommand(query, conn);
            SqlParameter[] parametros = new SqlParameter[2];
            parametros[0] = new SqlParameter("@user_name", SqlDbType.NVarChar);
            parametros[0].Value = TextBox_Users.Text;
            parametros[1] = new SqlParameter("@password", SqlDbType.NVarChar);
            parametros[1].Value = oDB.EncryptString(TextBox_Pass.Text, "secret");
            foreach (SqlParameter par in parametros)
            {
                cmd.Parameters.Add(par);
                par.Direction = System.Data.ParameterDirection.Input;
            }
            try
            {
                conn.Open();
                dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
                if (dr.HasRows)
                {
                    dr.Read();
                    Session.Add("id_utilizador", dr[0].ToString());
                    Session.Add("endereco", dr[1].ToString());
                    Session.Add("id_perfil", dr[2].ToString());
                    Session.Add("nome_utilizador", dr[3].ToString());
                    Session.Add("localidade", dr[6].ToString());
                    Session.Add("apto", dr[7].ToString());
                    Session.Add("idPredioMorador", dr[8].ToString());
                    conn.Close();
                    if (Session["id_perfil"].ToString() == "1">//perfil morador
                    {
                        Response.Redirect("/Master5/MoradorPrincipal.aspx", false);
                    }
                    if (Session["id_perfil"].ToString() == "2">//perfil sindico
                    {
                        Response.Redirect("/Master3/SindicoPrincipal.aspx", false);
                    }
                    if (Session["id_perfil"].ToString() == "4">//perfil administrador
                    {
                        Response.Redirect("/Master6/AdminPrincipal1.aspx", false);
                    }
                }
            }
        }
    }
}
```

```

        } //end if try
        else
        {
            this.TextBox_Users.Text = " ";
            this.TextBox_Pass.Text = " ";
            MessageBox.Show("Username ou Password Inválido!");
            Response.Redirect("/Master1/MenuHome.aspx", false);
        }

    } //end try
    catch (Exception ex)
    {
        MessageBox.Show("" + ex.Message);
    }
}
}
}

```

Código para recuperação de senha

```

namespace SmartCondo
{
    public partial class Site1 : System.Web.UI.MasterPage
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
        security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            DateTime agora = DateTime.Now;
            String hoje = String.Format("{0:dddd, d MMMM}", agora);
            Session.Add("data", hoje);
            lblHora.Text = Session["data"].ToString();
        }

        protected void LinkBtnRecuperaPass_Click(object sender, EventArgs e)
        {
            Response.Redirect("/Master1/RecuperaSenha.aspx", false);
        }

        protected void Button_Login_Click1(object sender, EventArgs e)
        {
            string query = "SELECT * FROM View_Utilizador_Predio WHERE user_name = @user_name
            and password = @password";
            SqlDataReader dr;
            SqlConnection conn = new SqlConnection(@"Data
            Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
            Security=True;User Instance=True");
            SqlCommand cmd = new SqlCommand(query, conn);
            SqlParameter[] parametros = new SqlParameter[2];
            parametros[0] = new SqlParameter("@user_name", SqlDbType.NVarChar);
            parametros[0].Value = TextBox_Users.Text;
            parametros[1] = new SqlParameter("@password", SqlDbType.NVarChar);
            parametros[1].Value = oDB.EncryptString(TextBox_Pass.Text, "secret");
            foreach (SqlParameter par in parametros)
            {
                cmd.Parameters.Add(par);
                par.Direction = System.Data.ParameterDirection.Input;
            }
            try
            {
                conn.Open();
                dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
                if (dr.HasRows)
                {
                    dr.Read();
                    Session.Add("id_utilizador", dr[0].ToString());
                    Session.Add("endereco", dr[1].ToString());
                    Session.Add("id_perfil", dr[2].ToString());
                }
            }
            catch { }
        }
    }
}

```

```

        Session.Add("nome_utilizador", dr[3].ToString());
        Session.Add("localidade", dr[6].ToString());
        Session.Add("apto", dr[7].ToString());
        Session.Add("idPredioMorador", dr[8].ToString());
        conn.Close();
        if (Session["id_perfil"].ToString() == "1")//perfil morador
        {
            Response.Redirect("/Master5/MoradorPrincipal.aspx", false);
        }
        if (Session["id_perfil"].ToString() == "2")//perfil sindico
        {
            Response.Redirect("/Master3/SindicoPrincipal.aspx", false);
        }
        if (Session["id_perfil"].ToString() == "4")//perfil administrador
        {
            Response.Redirect("/Master6/AdminPrincipal1.aspx", false);
        }

    } //end if try
    else
    {
        this.TextBox_Users.Text = " ";
        this.TextBox_Pass.Text = " ";
        MessageBox.Show("Username ou Password Inválido!");
        Response.Redirect("/Master1/MenuHome.aspx", false);
    }

} //end try
catch (Exception ex)
{
    MessageBox.Show("" + ex.Message);
}
}
}
}

```

Código para cadastro da área privada

```

namespace SmartCondo
{
    public partial class WebForm13 : System.Web.UI.Page
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
        security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["condo"] != null)
            {
                Label1.Text = Session["condo"].ToString();
                Label2.Text = Session["condo"].ToString();
            }
            else
            {
                MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
                autenticação no sistema.");
                Response.Redirect("/Master1/MenuHome.aspx", false);
            }
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            string predio = Session["idPredio"].ToString();

            string query = "SELECT * FROM View_AreaPrivada WHERE id_predio = @idPredio and
            descricao_itemAreaPrivada = @AreaPrivada";
            SqlDataReader dr;
        }
    }
}

```

```

        SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
        SqlCommand cmd = new SqlCommand(query, conn);
        SqlParameter[] parametros = new SqlParameter[2];
        parametros[0] = new SqlParameter("@idPredio", SqlDbType.Int);
        parametros[0].Value = Convert.ToInt32(Session["idPredio"].ToString());
        parametros[1] = new SqlParameter("@AreaPrivada", SqlDbType.NVarChar);
        parametros[1].Value = DropDownList1.SelectedItem.Text;
        foreach (SqlParameter par in parametros)
        {
            cmd.Parameters.Add(par);
            par.Direction = System.Data.ParameterDirection.Input;
        }
        try
        {
            conn.Open();
            dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
            if (dr.HasRows)
            {
                MessageBox.Show("Área Privada já cadastrada no condomínio, insira novamente
os dados.");
                conn.Close();
                Response.Redirect("/Master2/CadastroAreaPrivada.aspx", false);
            }
            else
            {
                oDB.cadastraAreaPrivada(DropDownList1.SelectedValue.ToString(), predio,
TextBox1.Text, TextBox2.Text);
                oDB.atualizaPredioPossuiAreaPrivada(Session["idPredio"].ToString(),
"Possui");
                MessageBox.Show("Área Privada inserida com sucesso!");
                Response.Redirect("/Master2/CadastroAreaPrivada.aspx", false);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("" + ex.Message);
        }
    }
}
}

```

Código para cadastro de utilizador

```

namespace SmartCondo
{
    public partial class WebForm6 : System.Web.UI.Page
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["condo"] != null)
            {
                Label1.Text = Session["condo"].ToString();
                Label2.Text = Session["condo"].ToString();
            }
            else
            {
                MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
autenticação no sistema.");
                Response.Redirect("/Master1/MenuHome.aspx", false);
            }
        }

        protected void Button1_Click(object sender, EventArgs e)
        {

```

```

Classes.MailSender email = new Classes.MailSender();
int senhaAleat = CalculosBasicos.Instance.GerarNumAleat();
string userName = TextBoxName.Text.Substring(0, 4);

string query = "SELECT * FROM Utilizador WHERE id_predio = @id_predio AND apto =
@apto OR email = @email";
SqlDataReader dr;
SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
SqlCommand cmd = new SqlCommand(query, conn);
SqlParameter[] parametros = new SqlParameter[3];
parametros[0] = new SqlParameter("@id_predio", SqlDbType.Int);
parametros[0].Value = Convert.ToInt32(Session["idPredio"].ToString());
parametros[1] = new SqlParameter("@apto", SqlDbType.NVarChar);
parametros[1].Value = TextBoxApto.Text;
parametros[2] = new SqlParameter("@email", SqlDbType.NVarChar);
parametros[2].Value = TextBoxEmail.Text;
foreach (SqlParameter par in parametros)
{
    cmd.Parameters.Add(par);
    par.Direction = System.Data.ParameterDirection.Input;
}
try
{
    conn.Open();
    dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
    if (dr.HasRows)
    {
        MessageBox.Show("Utilizador já existente nesse condomínio! Insira os dados
novamente.");
        Response.Redirect("/Master2/CadastroUser.aspx", false);
    }
    //end if try
    else
    {
        try
        {
            string condo = Session["idPredio"].ToString();
            string PassEncrypt = oDB.EncryptString(senhaAleat.ToString(),
"secret");
            DateTime agora = DateTime.Now;
            string statusCadastro = "Ativo";
            oDB.criaUtilizador(condo, DropDownListPerfil.SelectedValue.ToString(),
TextBoxName.Text, TextBoxApto.Text, TextBoxContacto.Text, TextBoxEmail.Text, userName,
PassEncrypt, agora, statusCadastro);
            String AssuntoTexto = "SmartCondo - Aviso de cadastro do morador(a) " +
TextBoxName.Text;
            String DescricaoTexto = "Prezado(a) " + TextBoxName.Text + ";\n \n
Seguem abaixo seus dados para autenticação no sistema SmartCondo.\n \n Utilizador: " + userName
+ "\n Código de acesso: " + senhaAleat + "\n \n Para acessar a página SmartCondo clique aqui.\n
\n Atenciosamente,\n Equipe SmartCondo.";
            email.EnviaMail(TextBoxEmail.Text, AssuntoTexto, DescricaoTexto);
            MessageBox.Show("Morador cadastrado com sucesso! Email com dados para
autenticação já enviado.");
            Response.Redirect("/Master2/CondoSelecionado.aspx", false);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Insucesso no cadastro do morador!: \n" + ex.Message);
        }
    }
}
//end try
catch (Exception ex)
{
    MessageBox.Show("" + ex.Message);
}
}

public sealed class CalculosBasicos
{
    private CalculosBasicos() { }
    private static readonly Random RANDOM = new Random();
    private static readonly CalculosBasicos INSTANCE = new CalculosBasicos();
}

```

```

        public static CalculosBasicos Instance
        {
            get
            {
                return INSTANCE;
            }
        }

        public int GerarNumAleat()
        {
            return RANDOM.Next(1000, 9999);
        }
    }
}
}

```

Código para cálculo da quota do condomínio

```

namespace SmartCondo
{
    public partial class WebForm42 : System.Web.UI.Page
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
        security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["condo"] != null)
            {
                Label2.Text = Session["condo"].ToString();
                lblFraser.Visible = false;
                lblMes.Visible = false;
                lblAno.Visible = false;
                DropDownList1.Visible = false;
                DropDownList2.Visible = false;
                GridView1.Visible = false;
                lblFraser2.Visible = false;
                lblValor.Visible = false;
                lblVenc.Visible = false;
                DatePicker1.Visible = false;
                Button1.Visible = false;
                Button2.Visible = false;
            }
            else
            {
                MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
                autenticação no sistema.");
                Response.Redirect("/Master1/MenuHome.aspx", false);
            }
        }

        protected void GridView2_RowCommand(object sender, GridViewCommandEventArgs e)
        {
            if (e.CommandName == "Select")
            {
                GridViewRow selectedRow =
                ((GridView)e.CommandSource).Rows[Convert.ToInt16(e.CommandArgument.ToString())];
                Session.Add("qMorador", selectedRow.Cells[1].Text);
                lblFraser.Visible = true;
                lblMes.Visible = true;
                lblAno.Visible = true;
                DropDownList1.Visible = true;
                DropDownList2.Visible = true;
                Button2.Visible = true;
            }
        }

        protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)

```



```

        {
            if (e.CommandName == "Select")
            {
                int valorFundo = 10;
                GridViewRow selectedRow =
                ((GridView)e.CommandSource).Rows[Convert.ToInt16(e.CommandArgument.ToString())];
                decimal valorTotal = Convert.ToDecimal(selectedRow.Cells[3].Text);
                int quantidadeMorador = Convert.ToInt32(Session["qMorador"].ToString());
                decimal valorCota = (Convert.ToDecimal(valorTotal / quantidadeMorador) +
                valorFundo);

                Session.Add("valorCota", valorCota);
                lblValor.Text = valorCota.ToString("0.00");
                lblFrase2.Visible = true;
                lblValor.Visible = true;
                lblVenc.Visible = true;
                DatePicker1.Visible = true;
                Button1.Visible = true;
                lblFrase.Visible = true;
                lblMes.Visible = true;
                lblAno.Visible = true;
                DropDownList1.Visible = true;
                DropDownList2.Visible = true;
                Button2.Visible = true;
                GridView1.Visible = true;
            }
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            string mesRef = DropDownList1.SelectedValue.ToString();
            string anoRef = DropDownList2.SelectedValue.ToString();
            string mesVenc = DatePicker1.SelectedDate.Month.ToString();
            string anoVenc = DatePicker1.SelectedDate.Year.ToString();

            int m1 = Convert.ToInt32(mesRef);
            int a1 = Convert.ToInt32(anoRef);
            int m2 = Convert.ToInt32(mesVenc);
            int a2 = Convert.ToInt32(anoVenc);

            if ((m1 < m2) & (a1 == a2)) || ((m1 > m2) & (a1 < a2))
            {
                oDB.cadastraCotaCondo(Session["idPredio"].ToString(),
                DropDownList1.SelectedItem.Text, DropDownList2.SelectedValue.ToString(),
                Session["valorCota"].ToString(), DatePicker1.SelectedDate.ToString());

                ///////////////////////////////////
                string query1 = "SELECT * FROM CotaCondo WHERE id_predio = @idPredio and
                mes_ref = @mesRef and ano_ref = @anoRef";
                SqlDataReader dr1;
                SqlConnection conn1 = new SqlConnection(@"Data
                Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
                Security=True;User Instance=True");
                SqlCommand cmd1 = new SqlCommand(query1, conn1);
                SqlParameter[] parametros1 = new SqlParameter[3];
                parametros1[0] = new SqlParameter("@idPredio", SqlDbType.Int);
                parametros1[0].Value = Convert.ToInt32(Session["idPredio"].ToString());
                parametros1[1] = new SqlParameter("@mesRef", SqlDbType.NVarChar);
                parametros1[1].Value = DropDownList1.SelectedItem.Text;
                parametros1[2] = new SqlParameter("@anoRef", SqlDbType.NVarChar);
                parametros1[2].Value = DropDownList2.SelectedValue.ToString();

                foreach (SqlParameter par1 in parametros1)
                {
                    cmd1.Parameters.Add(par1);
                    par1.Direction = System.Data.ParameterDirection.Input;
                }
                try
                {
                    conn1.Open();
                    dr1 = cmd1.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
                    if (dr1.HasRows)
                    {
                        dr1.Read();
                        Session.Add("idCotaCondo", dr1[0].ToString());

                        conn1.Close();
                    }
                }
            }
        }
    }
}

```

```

        } //end if try

    } //end try
    catch (Exception ex)
    {
        MessageBox.Show("" + ex.Message);
    }

    ///////////////////////////////////

    string query2 = "SELECT * FROM View_BuscaCota WHERE id_cotaCondo =
@idCotaCondo";
    SqlDataReader dr2;
    SqlConnection conn2 = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
    SqlCommand cmd2 = new SqlCommand(query2, conn2);
    SqlParameter[] parametros2 = new SqlParameter[1];
    parametros2[0] = new SqlParameter("@idCotaCondo", SqlDbType.Int);
    parametros2[0].Value = Convert.ToInt32(Session["idCotaCondo"].ToString());

    foreach (SqlParameter par2 in parametros2)
    {
        cmd2.Parameters.Add(par2);
        par2.Direction = System.Data.ParameterDirection.Input;
    }
    try
    {
        conn2.Open();
        dr2 = cmd2.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
        if (dr2.HasRows)
        {
            while (dr2.Read())
            {
                Session.Add("idPredioCota", dr2[1].ToString());
                Session.Add("idUtilizadorCota", dr2[2].ToString());
                oDB.cadastraPgtoCotaUtilizador(Session["idCotaCondo"].ToString(),
Session["idPredioCota"].ToString(), Session["idUtilizadorCota"].ToString(), "A Pagar");
            }
            conn2.Close();
        }
    } //end if try

    } //end try
    catch (Exception ex)
    {
        MessageBox.Show("" + ex.Message);
    }

    ///////////////////////////////////

    MessageBox.Show("Cota lançada com sucesso!!");
    Response.Redirect("/Master2/CondoSelecionado.aspx", false);
}
else
{
    MessageBox.Show("Datas de vencimento incoerente. Favor inserir novamente os
dados.");
    Response.Redirect("/Master2/CalculaCota.aspx", false);
}

}

protected void Button2_Click(object sender, EventArgs e)
{
    string query = "SELECT * FROM CotaCondo WHERE id_predio = @idPredio and mes_ref =
@mesRef and ano_ref = @anoRef";
    SqlDataReader dr;
    SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
    SqlCommand cmd = new SqlCommand(query, conn);
    SqlParameter[] parametros = new SqlParameter[3];
    parametros[0] = new SqlParameter("@idPredio", SqlDbType.Int);

```

```

parametros[0].Value = Convert.ToInt32(Session["idPredio"].ToString());
parametros[1] = new SqlParameter("@mesRef", SqlDbType.NVarChar);
parametros[1].Value = DropDownList1.SelectedItem.Text;
parametros[2] = new SqlParameter("@anoRef", SqlDbType.NVarChar);
parametros[2].Value = DropDownList2.SelectedItem.Text;

foreach (SqlParameter par in parametros)
{
    cmd.Parameters.Add(par);
    par.Direction = System.Data.ParameterDirection.Input;
}
try
{
    conn.Open();
    dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
    if (dr.HasRows)
    {
        MessageBox.Show("Cota " + DropDownList1.SelectedItem.Text + "/" +
DropDownList2.SelectedItem.Text + " já cadastrada. Insira novamente os dados.");
        conn.Close();
        Response.Redirect("/Master2/CalculaCota.aspx", false);
    }
    //end if try
    else
    {
        Session.Add("mes", DropDownList1.SelectedItem.Text);
        lblFrase.Visible = true;
        lblMes.Visible = true;
        lblAno.Visible = true;
        DropDownList1.Visible = true;
        DropDownList2.Visible = true;
        Button2.Visible = true;
        GridView1.Visible = true;
    }
}
//end try
catch (Exception ex)
{
    MessageBox.Show("" + ex.Message);
}
}
}
}
}

```

Código para cancelar condomínio

```

namespace SmartCondo
{
    public partial class WebForm3 : System.Web.UI.Page
    {
        static string strCon =
ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["condo"] != null)
            {
                lblCondo.Text = Session["condo"].ToString();
                Label2.Text = Session["condo"].ToString();
            }
            else
            {
                MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
autenticação no sistema.");
                Response.Redirect("/Master1/MenuHome.aspx", false);
            }
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            DateTime agora = DateTime.Now;
            oDB.cancelaCondo(Session["idPredio"].ToString(), "Desativo", TextBox1.Text, agora);
            MessageBox.Show("Condominio cancelado com sucesso!");
        }
    }
}

```

```

        oDB.cancelaMoradorPosCondo(Session["idPredio"].ToString(), "Desativo", "Condomínio
desativado.", agora);
        MessageBox.Show("Moradores desativados com sucesso!");
        Response.Redirect("/Master6/AdminPrincipal1.aspx", false);
    }
}
}

```

Código para cancelar morador

```

namespace SmartCondo
{
    public partial class WebForm16 : System.Web.UI.Page
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["condo"] != null)
            {
                Label1.Text = Session["condo"].ToString();
                Label2.Text = Session["condo"].ToString();
            }
            else
            {
                MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
autenticação no sistema.");
                Response.Redirect("/Master1/MenuHome.aspx", false);
            }
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            DateTime agora = DateTime.Now;
            string cadastro = "Desactivo";
            oDB.cancelaMorador(DropDownList1.SelectedValue.ToString(), cadastro, TextBox1.Text,
agora, "1");
            MessageBox.Show("Morador cancelado com sucesso!");
            Response.Redirect("/Master2/CondoSelecioneado.aspx", false);
        }
    }
}

```

Código para lançar conta do condomínio

```

namespace SmartCondo
{
    public partial class WebForm22 : System.Web.UI.Page
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["condo"] != null)
            {
                Label1.Text = Session["condo"].ToString();
                Label2.Text = Session["condo"].ToString();
            }
            else
            {
                MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
autenticação no sistema.");
                Response.Redirect("/Master1/MenuHome.aspx", false);
            }
        }
    }
}

```

```

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        string query = "SELECT * FROM CotaCondo WHERE id_predio = @idPredio and mes_ref = @mesRef and ano_ref = @anoRef";
        SqlDataReader dr;
        SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
        SqlCommand cmd = new SqlCommand(query, conn);
        SqlParameter[] parametros = new SqlParameter[3];
        parametros[0] = new SqlParameter("@idPredio", SqlDbType.Int);
        parametros[0].Value = Convert.ToInt32(Session["idPredio"].ToString());
        parametros[1] = new SqlParameter("@mesRef", SqlDbType.NVarChar);
        parametros[1].Value = DropDownList1.SelectedItem.Text;
        parametros[2] = new SqlParameter("@anoRef", SqlDbType.NVarChar);
        parametros[2].Value = DropDownList2.SelectedItem.Text;

        foreach (SqlParameter par in parametros)
        {
            cmd.Parameters.Add(par);
            par.Direction = System.Data.ParameterDirection.Input;
        }
        try
        {
            conn.Open();
            dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
            if (dr.HasRows)
            {
                MessageBox.Show("Cota " + DropDownList1.SelectedItem.Text + "/" +
                DropDownList2.SelectedItem.Text + " já lançada. Não é possível inserir mais despesas nesse
                período.");
                conn.Close();
                Response.Redirect("/Master2/LancamentoConta.aspx", false);
            }
            //end if try
        }
        else
        {
            string mesRef = DropDownList1.SelectedValue.ToString();
            string anoRef = DropDownList2.SelectedValue.ToString();
            string mesVenc = DatePicker1.SelectedDate.Month.ToString();
            string anoVenc = DatePicker1.SelectedDate.Year.ToString();

            int m1 = Convert.ToInt32(mesRef);
            int a1 = Convert.ToInt32(anoRef);
            int m2 = Convert.ToInt32(mesVenc);
            int a2 = Convert.ToInt32(anoVenc);

            string condo = Session["idPredio"].ToString();

            if ((m1 < m2) & (a1 == a2)) || ((m1 > m2) & (a1 < a2))
            {
                string query1 = "SELECT * FROM Conta WHERE id_predio = @idPredio and
                factura = @factura";
                SqlDataReader dr1;
                SqlConnection conn1 = new SqlConnection(@"Data
                Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
                Security=True;User Instance=True");
                SqlCommand cmd1 = new SqlCommand(query1, conn1);
                SqlParameter[] parametros1 = new SqlParameter[2];
                parametros1[0] = new SqlParameter("@idPredio", SqlDbType.Int);
                parametros1[0].Value = Convert.ToInt32(Session["idPredio"].ToString());
                parametros1[1] = new SqlParameter("@factura", SqlDbType.NVarChar);
                parametros1[1].Value = TextBoxFactura.Text;
                foreach (SqlParameter par1 in parametros1)
                {
                    cmd1.Parameters.Add(par1);
                    par1.Direction = System.Data.ParameterDirection.Input;
                }
                try
                {
                    conn1.Open();
                    dr1 =
                    cmd1.ExecuteReader(System.Data.CommandBehavior.CloseConnection);

```



```

    }
    else
    {
        MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
autenticação no sistema.");
        Response.Redirect("/Master1/MenuHome.aspx", false);
    }
}

protected void ButtonInsereConta_Click(object sender, EventArgs e)
{
    Response.Redirect("/Master2/ListaConta.aspx", false);
}

protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "Select")
    {
        GridViewRow selectedRow =
((GridView)e.CommandSource).Rows[Convert.ToInt16(e.CommandArgument.ToString())];
        string idConta = selectedRow.Cells[0].Text;
        Session.Add("idConta", idConta);
        string enderecoCondo = selectedRow.Cells[1].Text;
        Session.Add("enderecoCondo", enderecoCondo);
        string tipoConta = selectedRow.Cells[2].Text;
        string vencimento = selectedRow.Cells[4].Text;
        string valorConta = selectedRow.Cells[5].Text;

        string query = "SELECT * FROM Predio WHERE id_predio = @id_predio";
        SqlDataReader dr;
        SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
        SqlCommand cmd = new SqlCommand(query, conn);
        SqlParameter[] parametros = new SqlParameter[1];
        parametros[0] = new SqlParameter("@id_predio", SqlDbType.Int);
        parametros[0].Value = Session["idPredio"].ToString();

        foreach (SqlParameter par in parametros)
        {
            cmd.Parameters.Add(par);
            par.Direction = System.Data.ParameterDirection.Input;
        }
        try
        {
            conn.Open();
            dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
            if (dr.HasRows)
            {
                dr.Read();
                Session.Add("valorAnteriorFundo", dr[8].ToString());

                conn.Close();

                //end if try
            }
        }
        //end try
        catch (Exception ex)
        {
            MessageBox.Show("" + ex.Message);
        }

        if ((Convert.ToDecimal(Session["valorAnteriorFundo"].ToString())) <
(Convert.ToDecimal(valorConta)))
        {
            MessageBox.Show("O condomínio não possui saldo suficiente para o pagamento
dessa conta. Saldo condomínio: " + Session["valorAnteriorFundo"].ToString() + ". Saldo
necessário para pagamento: " + valorConta + ".");
            Response.Redirect("/Master2/CondoSelecionado.aspx", false);
        }
    }
    else
    {

```

```

        decimal valorActual =
((Convert.ToDecimal(Session["valorAnteriorFundo"].ToString())) -
(Convert.ToDecimal(valorConta)));
        Session.Add("valorActualFundo", valorActual.ToString());
        Label3.Text = tipoConta;
        Label8.Text = valorConta;
        Label6.Text = vencimento;
        Label10.Text = Session["valorAnteriorFundo"].ToString();
        Label12.Text = Session["valorActualFundo"].ToString();
        Label3.Visible = true;
        Label13.Visible = true;
        Label14.Visible = true;
        Label11.Visible = true;
        Label9.Visible = true;
        Label8.Visible = true;
        Label10.Visible = true;
        Label12.Visible = true;
        Label4.Visible = true;
        Label5.Visible = true;
        Label7.Visible = true;
        Label6.Visible = true;
        Button1.Visible = true;
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    DateTime agora = DateTime.Now;
    oDB.actualizaFundoPredio(Session["idPredio"].ToString(),
Session["valorActualFundo"].ToString());
    oDB.actualizaStatusContaPredio(Session["idConta"].ToString(),
Session["idPredio"].ToString(), "2", agora.ToString());
    MessageBox.Show("Pagamento realizado com sucesso. Saldo do condomínio: " +
Session["valorActualFundo"].ToString());
    Response.Redirect("/Master2/CondoSelecioneado.aspx", false);
}
}
}

```

Código para aprovar ou reprovar reserva de área privada

```

namespace SmartCondo
{
    public partial class WebForm37 : System.Web.UI.Page
    {
        static string strCon =
ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            Label2.Text = Session["endereco"].ToString();
            Label3.Text = Session["nome utilizador"].ToString();
            Label5.Text = Session["endereco"].ToString();
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            foreach (GridViewRow row in GridView1.Rows)
            {
                CheckBox ch = (CheckBox)row.FindControl("CheckBox2");
                if (ch != null)
                {
                    if (ch.Checked)
                    {
                        string idReserva = row.Cells[0].Text;
                        string infoEstado = "Reserva Aprovada";
                        oDB.actualizaEstadoReserva(idReserva, infoEstado);
                    }
                }
            }
        }
    }
}

```



```

    }
    //MessageBox.Show("Solicitações aprovadas com sucesso.");
    Response.Redirect("/Master3/GestaoReservaSindico.aspx", false);
}

protected void CheckBox2_CheckedChanged(object sender, EventArgs e)
{
}

protected void Button2_Click(object sender, EventArgs e)
{
    foreach (GridViewRow row in GridView1.Rows)
    {
        CheckBox ch = (CheckBox)row.FindControl("CheckBox2");
        if (ch != null)
        {
            if (ch.Checked)
            {
                string idReserva = row.Cells[0].Text;
                string infoEstado = "Reserva Reprovada";
                oDB.atualizaEstadoReserva(idReserva, infoEstado);
            }
        }
    }
    //MessageBox.Show("Solicitações aprovadas com sucesso.");
    Response.Redirect("/Master3/GestaoReservaSindico.aspx", false);
}

}

}

```

Código para convocar assembleia

```

namespace SmartCondo
{
    public partial class WebForm8 : System.Web.UI.Page
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
        security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            Label2.Text = Session["endereco"].ToString();
            Label3.Text = Session["nome_utilizador"].ToString();
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
        }

        protected void Button1_Click1(object sender, EventArgs e)
        {
            Classes.MailSender email = new Classes.MailSender();
            String AssuntoTexto = "Convocação para Assembléia do condomínio " +
            Session["endereco"].ToString() + ".";

            string query = "SELECT *FROM Utilizador WHERE id_predio =@idPredio";
            SqlDataReader dr;
            SqlConnection conn = new SqlConnection(@"Data
            Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
            Security=True;User Instance=True");
            SqlCommand cmd = new SqlCommand(query, conn);
            SqlParameter[] parametros = new SqlParameter[1];
            parametros[0] = new SqlParameter("@idPredio", SqlDbType.Int);
            parametros[0].Value = Convert.ToInt32(Session["idPredioMorador"].ToString());
            foreach (SqlParameter par in parametros)

```

```

        {
            cmd.Parameters.Add(par);
            par.Direction = System.Data.ParameterDirection.Input;
        }
        try
        {
            conn.Open();
            dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    Session.Add("utilizador", dr[0].ToString());
                    Session.Add("email", dr[6].ToString());
                    Session.Add("nome", dr[3].ToString());
                    try
                    {
                        String DescricaoTexto = "Prezado(a) " + Session["nome"].ToString()
+ ",\n \n Serve o presente para a convocação da Assembléia do condomínio " +
Session["endereco"].ToString() + ".\n \n A decorrer no dia: " +
DatePicker1.SelectedDate.ToShortDateString() + "\n Horário: " + TextBoxHora.Text + "\n Assunto
a ser abordado: " + TextBoxAssunto.Text + "\n Local: " + TextBoxLocal.Text + "\n \n
Atenciosamente,\n Síndico SmartCondo.";
                        email.EnviaMail(Session["email"].ToString(), AssuntoTexto,
DescricaoTexto);
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Erro ao enviar o email.: " + ex);
                        Response.Redirect("/Master3/SindicoPrincipal.aspx", false);
                    }
                }
                conn.Close();
                MessageBox.Show("Email enviado a todos os condôminos.");
            }

        } //end if try

    } //end try
    catch (Exception ex)
    {
        MessageBox.Show("" + ex.Message);
    }
}
}
}

```

Código para morador pagar cota

```

namespace SmartCondo
{
    public partial class WebForm36 : System.Web.UI.Page
    {
        static string strCon =
ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["nome_utilizador"] != null)
            {
                Label2.Text = Session["endereco"].ToString();
                Label3.Text = Session["nome_utilizador"].ToString();
                lblEntidade.Visible = false;
                lblEntResult.Visible = false;
                lblRef.Visible = false;
                lblRefResult.Visible = false;
                lblValor.Visible = false;
            }
        }
    }
}

```

```

        lblValorResult.Visible = false;
        btnPagarCota.Visible = false;
    }
    else
    {
        MessageBox.Show("Sessão encerrada por tempo, por favor faça novamente a sua
autenticação no sistema.");
        Response.Redirect("/Master1/MenuHome.aspx", false);
    }
    if (Session["contaAtrasada"] != null)
    {
        Label4.Text = "AVISO! Existe conta pendente para pagamento, por favor
regularize a sua situação.";
    }
    else
    {
        Label4.Visible = false;
    }
}

protected void GridView2_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "Select")
    {
        GridViewRow selectedRow =
((GridView)e.CommandSource).Rows[Convert.ToInt16(e.CommandArgument.ToString())];
        lblValorResult.Text = selectedRow.Cells[5].Text;

        int RefMultibanco = CalculosBasicos.Instance.GerarRefMultibanco();
        lblRefResult.Text = RefMultibanco.ToString();

        lblEntidade.Visible = true;
        lblEntResult.Visible = true;
        lblRef.Visible = true;
        lblRefResult.Visible = true;
        lblValor.Visible = true;
        lblValorResult.Visible = true;
        btnPagarCota.Visible = true;

        Session.Add("valorMoradorPagaCota", selectedRow.Cells[5].Text);
        Session.Add("idCotaCondo", selectedRow.Cells[0].Text);
    }
}

public sealed class CalculosBasicos
{
    private CalculosBasicos() { }
    private static readonly Random RANDOM = new Random();
    private static readonly CalculosBasicos INSTANCE = new CalculosBasicos();

    public static CalculosBasicos Instance
    {
        get
        {
            return INSTANCE;
        }
    }

    public int GerarRefMultibanco()
    {
        return RANDOM.Next(10000000, 999999999);
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    string query = "SELECT * FROM Predio WHERE id_predio = @id_predio";
    SqlDataReader dr;
    SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
    SqlCommand cmd = new SqlCommand(query, conn);
    SqlParameter[] parametros = new SqlParameter[1];
    parametros[0] = new SqlParameter("@id_predio", SqlDbType.Int);

```

```

        parametros[0].Value = Session["idPredioMorador"].ToString();

        foreach (SqlParameter par in parametros)
        {
            cmd.Parameters.Add(par);
            par.Direction = System.Data.ParameterDirection.Input;
        }
        try
        {
            conn.Open();
            dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
            if (dr.HasRows)
            {
                dr.Read();
                Session.Add("valorAnteriorFundo", dr[8].ToString());

                conn.Close();

                } //end if try
            } //end try
        catch (Exception ex)
        {
            MessageBox.Show("" + ex.Message);
        }

        decimal valorActual =
        (Convert.ToDecimal(Session["valorMoradorPagaCota"].ToString())) +
        (Convert.ToDecimal(Session["valorAnteriorFundo"].ToString()));

        oDB.moradorPagaCota(valorActual.ToString(), Session["idPredioMorador"].ToString());
        oDB.actualizaStatusPgtoCota(Session["id_utilizador"].ToString(),
        Session["idCotaCondo"].ToString(), Session["idPredioMorador"].ToString(), "Paga");
        MessageBox.Show("Cota paga com sucesso!");
        Response.Redirect("/Master3/SindicoPrincipal.aspx", false);
    }
}
}

```

Código para reservar área privada

```

namespace SmartCondo
{
    public partial class WebForm35 : System.Web.UI.Page
    {
        static string strCon =
        ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
        security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        DataTable areaPrivadaSelec = new DataTable(); //tabela temporária (gridview)

        protected void Page_Load(object sender, EventArgs e)
        {
            Label2.Text = Session["endereco"].ToString();
            Label3.Text = Session["nome_utilizador"].ToString();
            Label5.Text = Session["endereco"].ToString();
            areaPrivadaSelec.Columns.Add("Tipo_Area");
            areaPrivadaSelec.Columns.Add("Tempo_Reserva");
            areaPrivadaSelec.Columns.Add("Valor");
        }

        protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
        {
            if (e.CommandName == "Select")
            {
                GridViewRow selectedRow =
                ((GridView)e.CommandSource).Rows[Convert.ToInt16(e.CommandArgument.ToString())];
                string idArea = selectedRow.Cells[0].Text;
                Session.Add("idArea", idArea);
            }
        }
    }
}

```

```

        string idDescArea = selectedRow.Cells[1].Text;
        string tempo = selectedRow.Cells[2].Text;
        string preco = selectedRow.Cells[3].Text;

        DataRow drNewRow = areaPrivadaSelec.NewRow();

        drNewRow["Tipo_Area"] = idDescArea;
        drNewRow["Tempo_Reserva"] = tempo;
        drNewRow["Valor"] = preco;

        areaPrivadaSelec.Rows.Add(drNewRow);
        areaPrivadaSelec.AcceptChanges();

        GridView2.DataSource = areaPrivadaSelec;
        GridView2.DataBind();
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        string query = "SELECT * FROM Reserva WHERE id_detalheAreaPrivada = @idAreaPrivada
and diaReserva = @diaReserva";
        SqlDataReader dr;
        SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
        SqlCommand cmd = new SqlCommand(query, conn);
        SqlParameter[] parametros = new SqlParameter[2];
        parametros[0] = new SqlParameter("@idAreaPrivada", SqlDbType.Int);
        parametros[0].Value = Convert.ToInt32(Session["idArea"].ToString());
        parametros[1] = new SqlParameter("@diaReserva", SqlDbType.DateTime);
        parametros[1].Value = Convert.ToDateTime (DatePicker1.SelectedDate.ToString());
        foreach (SqlParameter par in parametros)
        {
            cmd.Parameters.Add(par);
            par.Direction = System.Data.ParameterDirection.Input;
        }
        try
        {
            conn.Open();
            dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
            if (dr.HasRows)
            {
                MessageBox.Show("Já existe reserva solicitada para esse dia.");
                conn.Close();
                Response.Redirect("/Master3/ReservaAreaMorador.aspx", false);

            }
            //end if try
            else
            {
                DateTime agora = DateTime.Now;
                DateTime dtReserva =
Convert.ToDateTime (DatePicker1.SelectedDate.ToString());

                if (agora.CompareTo(dtReserva) == 1)
                {
                    //agora maior que a data2
                    MessageBox.Show("Incoerência na data solicitada! Data já passada.");
                    Response.Redirect("/Master3/ReservaAreaMorador.aspx", false);
                }
                else
                {
                    oDB.cadastraReserva(Session["id_utilizador"].ToString(),
Session["idArea"].ToString(), "Aguarda Aprovação", DatePicker1.SelectedDate.ToString(),
Session["data"].ToString());
                    MessageBox.Show("Reserva solicitada. Por favor aguardar aprovação.");
                    Response.Redirect("/Master3/SindicoPrincipal.aspx", false);
                }
            }
        }
        //end try
        catch (Exception ex)
        {
            MessageBox.Show("" + ex.Message);
        }
    }
}

```

```

    }

}

```

Código para cadastro do condomínio

```

namespace SmartCondo
{
    public partial class WebForm5 : System.Web.UI.Page
    {
        static string strCon =
ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString + ";integrated
security=SSPI;persist security info=True";
        Classes.BD oDB = new Classes.BD(strCon);

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void ButtonInsereCondo_Click(object sender, EventArgs e)
        {
            DateTime agora = DateTime.Now;
            string query = "SELECT * FROM Predio WHERE endereco = @endereco";
            SqlDataReader dr;
            SqlConnection conn = new SqlConnection(@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BaseDados.mdf;Integrated
Security=True;User Instance=True");
            SqlCommand cmd = new SqlCommand(query, conn);
            SqlParameter[] parametros = new SqlParameter[1];
            parametros[0] = new SqlParameter("@endereco", SqlDbType.NVarChar);
            parametros[0].Value = TextBoxEnd.Text;
            foreach (SqlParameter par in parametros)
            {
                cmd.Parameters.Add(par);
                par.Direction = System.Data.ParameterDirection.Input;
            }
            try
            {
                conn.Open();
                dr = cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);

                if (dr.HasRows)
                {
                    this.TextBoxEnd.Text = " ";
                    this.TextBoxLoc.Text = " ";
                    this.TextBoxCod.Text = " ";
                    MessageBox.Show("Endereço já existente na base de dados!");
                }
                else
                {
                    oDB.criaCondominio(TextBoxEnd.Text, TextBoxCod.Text, TextBoxLoc.Text,
agora, "Ativo", "0", "Não");
                    MessageBox.Show("Condomínio cadastrado com sucesso!");
                    Response.Redirect("/Master6/AdminPrincipall.aspx", false);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Insucesso no registo do condomínio!: \n" + ex.Message);
            }
        }

        protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
        {
            Response.Redirect("/Master2/CondosCadastrados.aspx", false);
        }
    }
}

```

Código da classe utilizada para envio de emails

```
namespace Classes
{
    public class MailSender
    {
        public bool EnviaMail(string destinatario, string assunto, string conteudo)
        {
            string pGmailEmail = "SmartCondoBR@gmail.com";
            string pGmailPassword = "smart123";

            try
            {
                System.Web.Mail.MailMessage meuMail = new System.Web.Mail.MailMessage();
                meuMail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtpserver",
                "smtp.gmail.com");

                meuMail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtpserverport", "465");
                meuMail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendusing",
                "2");

                meuMail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate", "1");

                meuMail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendusername", pGmailEmail);

                meuMail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendpassword",
                pGmailPassword);

                meuMail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/smtpusessl",
                "true");

                meuMail.From = pGmailEmail;
                meuMail.To = destinatario;
                meuMail.Subject = assunto;
                meuMail.BodyFormat = System.Web.Mail.MailFormat.Text;
                meuMail.Body = conteudo;

                System.Web.Mail.SmtpMail.SmtpServer = "smtp.gmail.com:465";
                System.Web.Mail.SmtpMail.Send(meuMail);
                return true;
            }
            catch (Exception ex)
            {
                throw;
            }
        }
    }
}
```

Código da classe de acesso à base de dados com todos os métodos utilizado no sistema

```
namespace Classes
{
    public class BD
    {
        public BD()
        {
        }

        private string cn;

        public BD(string strConexao)
        {
            cn = strConexao;
        }

        protected SqlConnection getConexao()
        {
            SqlConnection retCon;
            retCon = new SqlConnection(cn);
        }
    }
}
```

```

        retCon.Open();
        return retCon;
    }

    protected void closeConexao(SqlConnection con)
    {
        con.Close();
        con = null;
    }

    public string EncryptString(string Message, string Passphrase)
    {
        byte[] Results;
        System.Text.UTF8Encoding UTF8 = new System.Text.UTF8Encoding();
        // Step 1. We hash the passphrase using MD5
        // We use the MD5 hash generator as the result is a 128 bit byte array
        // which is a valid length for the TripleDES encoder we use below
        MD5CryptoServiceProvider HashProvider = new MD5CryptoServiceProvider();
        byte[] TDESKey = HashProvider.ComputeHash(UTF8.GetBytes(Passphrase));
        // Step 2. Create a new TripleDESCryptoServiceProvider object
        TripleDESCryptoServiceProvider TDESAlgorithm = new
TripleDESCryptoServiceProvider();
        // Step 3. Setup the encoder
        TDESAlgorithm.Key = TDESKey;
        TDESAlgorithm.Mode = CipherMode.ECB;
        TDESAlgorithm.Padding = PaddingMode.PKCS7;
        // Step 4. Convert the input string to a byte[]
        byte[] DataToEncrypt = UTF8.GetBytes(Message);
        // Step 5. Attempt to encrypt the string
        try
        {
            ICryptoTransform Encryptor = TDESAlgorithm.CreateEncryptor();
            Results = Encryptor.TransformFinalBlock(DataToEncrypt, 0,
DataToEncrypt.Length);
        }
        finally
        {
            // Clear the TripleDes and Hashprovider services of any sensitive information
            TDESAlgorithm.Clear();
            HashProvider.Clear();
        }
        // Step 6. Return the encrypted string as a base64 encoded string
        return Convert.ToBase64String(Results);
    }

    public int criaCondominio(string nEndereco, string nCodigoPostal, string nLocalidade,
DateTime ndataCriacao, string statusContrato, string nfundo, string areaPrivada)
    {
        SqlConnection ligacao = getConexao();
        SqlCommand comando = new SqlCommand("insert into
Predio(endereco,codigo_postal,localidade, dataCriacao, contrato, fundo, possuiAreaPrivada)
values (@nEndereco, @nCodigoPostal, @nLocalidade, @ndataCriacao, @ncontrato, @nfundo,
@nareaPrivada)", ligacao);

        SqlParameter parameterEndereco = new SqlParameter("@nEndereco",
SqlDbType.NVarChar);
        parameterEndereco.Value = nEndereco;
        comando.Parameters.Add(parameterEndereco);

        SqlParameter parameterCodigoPostal = new SqlParameter("@nCodigoPostal",
SqlDbType.NVarChar);
        parameterCodigoPostal.Value = nCodigoPostal;
        comando.Parameters.Add(parameterCodigoPostal);

        SqlParameter parameterLocalidade = new SqlParameter("@nLocalidade",
SqlDbType.NVarChar);
        parameterLocalidade.Value = nLocalidade;
        comando.Parameters.Add(parameterLocalidade);

        DateTime data = Convert.ToDateTime(ndataCriacao);

        SqlParameter parameterDataCriacao = new SqlParameter("@ndataCriacao",
SqlDbType.DateTime);
        parameterDataCriacao.Value = ndataCriacao;
    }

```



```

        comando.Parameters.Add(parameterDataCriacao);

        SqlParameter parameterContrato = new SqlParameter("@ncontrato",
SqlDbType.NVarChar);
        parameterContrato.Value = statusContrato;
        comando.Parameters.Add(parameterContrato);

        SqlParameter parameterFundo = new SqlParameter("@nfundo", SqlDbType.Decimal);
        parameterFundo.Value = Convert.ToDecimal(nfundo);
        comando.Parameters.Add(parameterFundo);

        SqlParameter parameterAreaPrivada = new SqlParameter("@nareaPrivada",
SqlDbType.NVarChar);
        parameterAreaPrivada.Value = areaPrivada;
        comando.Parameters.Add(parameterAreaPrivada);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int criaUtilizador(string nEndereco, string nPerfil, string nNome, string nApto,
string nContacto, string nEmail, string nUser, string nPassword, DateTime ndataCriacao, string
nstatusCadastro)
    {

        SqlConnection ligacao = getConexao();

        SqlCommand comando = new SqlCommand("insert into
Utilizador(id_predio,id_perfil,nome_utilizador,apto,contacto,email,user_name,password,dataCriac
ao, cadastro) values (@nEndereco, @nPerfil, @nNome, @nApto, @nContacto, @nEmail, @nUser,
@nPassword, @ndataCriacao, @nstatusCadastro)", ligacao);

        SqlParameter parameterEndereco = new SqlParameter("@nEndereco", SqlDbType.Int);
        parameterEndereco.Value = Convert.ToInt32(nEndereco);
        comando.Parameters.Add(parameterEndereco);

        SqlParameter parameterPerfil = new SqlParameter("@nPerfil", SqlDbType.Int);
        parameterPerfil.Value = Convert.ToInt32(nPerfil);
        comando.Parameters.Add(parameterPerfil);

        SqlParameter parameterNome = new SqlParameter("@nNome", SqlDbType.NVarChar);
        parameterNome.Value = nNome;
        comando.Parameters.Add(parameterNome);

        SqlParameter parameterApto = new SqlParameter("@nApto", SqlDbType.NVarChar);
        parameterApto.Value = nApto;
        comando.Parameters.Add(parameterApto);

        SqlParameter parameterContacto = new SqlParameter("@nContacto", SqlDbType.Int);
        parameterContacto.Value = Convert.ToInt32(nContacto);
        comando.Parameters.Add(parameterContacto);

        SqlParameter parameterEmail = new SqlParameter("@nEmail", SqlDbType.NVarChar);
        parameterEmail.Value = nEmail;
        comando.Parameters.Add(parameterEmail);

        SqlParameter parameterUsername = new SqlParameter("@nUser", SqlDbType.NVarChar);
        parameterUsername.Value = nUser;
        comando.Parameters.Add(parameterUsername);

        SqlParameter parameterPassword = new SqlParameter("@nPassword",
SqlDbType.NVarChar);
        parameterPassword.Value = nPassword;
        comando.Parameters.Add(parameterPassword);

        DateTime data = Convert.ToDateTime(ndataCriacao);

        SqlParameter parameterDataCriacao = new SqlParameter("@ndataCriacao",
SqlDbType.DateTime);
        parameterDataCriacao.Value = data;
        comando.Parameters.Add(parameterDataCriacao);

        SqlParameter parameterStatusCadastro = new SqlParameter("@nstatusCadastro",
SqlDbType.NVarChar);
        parameterStatusCadastro.Value = nstatusCadastro;

```

```

        comando.Parameters.Add(parameterStatusCadastro);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int cancelaMorador(string nUtilizador, string nCadastro, string nInfo, DateTime
nDataFim, string idPerfil)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Utilizador] SET [id_perfil] = @idPerfil, [cadastro] =
@cadastro, [dataFim] = @dataFim, [info] = @info WHERE [id_utilizador] = @utilizador";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterIdPerfil = new SqlParameter("@idPerfil", SqlDbType.Int);
        parameterIdPerfil.Value = Convert.ToInt32(idPerfil);
        comando.Parameters.Add(parameterIdPerfil);

        SqlParameter parameterUtilizador = new SqlParameter("@utilizador", SqlDbType.Int);
        parameterUtilizador.Value = Convert.ToInt32(nUtilizador);
        comando.Parameters.Add(parameterUtilizador);

        SqlParameter parameterCadastro = new SqlParameter("@cadastro", SqlDbType.NVarChar);
        parameterCadastro.Value = nCadastro;
        comando.Parameters.Add(parameterCadastro);

        SqlParameter parameterInfo = new SqlParameter("@info", SqlDbType.NVarChar);
        parameterInfo.Value = nInfo;
        comando.Parameters.Add(parameterInfo);

        SqlParameter parameterDataFim = new SqlParameter("@dataFim", SqlDbType.DateTime);
        parameterDataFim.Value = nDataFim;
        comando.Parameters.Add(parameterDataFim);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int cancelaCondo(string nidPredio, string nContrato, string nInfo, DateTime
nDataFim)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Predio] SET [contrato] = @contrato, [dataFim] =
@dataFim, [info] = @info WHERE [id_predio] = @idPredio";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterPredio = new SqlParameter("@idPredio", SqlDbType.Int);
        parameterPredio.Value = nidPredio;
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterContrato = new SqlParameter("@contrato", SqlDbType.NVarChar);
        parameterContrato.Value = nContrato;
        comando.Parameters.Add(parameterContrato);

        SqlParameter parameterDataFim = new SqlParameter("@dataFim", SqlDbType.DateTime);
        parameterDataFim.Value = nDataFim;
        comando.Parameters.Add(parameterDataFim);

        SqlParameter parameterInfo = new SqlParameter("@info", SqlDbType.NVarChar);
        parameterInfo.Value = nInfo;
        comando.Parameters.Add(parameterInfo);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int lancaConta(string tipoConta, string statusConta, string condo, string
factura, string mesRef, string anoRef, string valorFact, string venc, string obs)
    {

```

```

        SqlConnection ligacao = getConexao();

        SqlCommand comando = new SqlCommand("insert into
Conta(id_tipoConta,id_statusConta,id_predio,factura,mes_ref,ano_ref,valor,vencimento,info)
values
(@nid_tipoConta,@nid_statusConta,@nid_predio,@nfactura,@nmes_ref,@nano_ref,@nvalor,@nvencimento
,@ninfo)", ligacao);

        SqlParameter parameterTipoConta = new SqlParameter("@nid_tipoConta",
SqlDbType.Int);
        parameterTipoConta.Value = Convert.ToInt32(tipoConta);
        comando.Parameters.Add(parameterTipoConta);

        SqlParameter parameterStatusConta = new SqlParameter("@nid_statusConta",
SqlDbType.Int);
        parameterStatusConta.Value = Convert.ToInt32(statusConta);
        comando.Parameters.Add(parameterStatusConta);

        SqlParameter parameterPredio = new SqlParameter("@nid_predio", SqlDbType.Int);
        parameterPredio.Value = Convert.ToInt32(condo);
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterFactura = new SqlParameter("@nfactura", SqlDbType.Int);
        parameterFactura.Value = Convert.ToInt32(factura);
        comando.Parameters.Add(parameterFactura);

        SqlParameter parameterMesRef = new SqlParameter("@nmes_ref", SqlDbType.NVarChar);
        parameterMesRef.Value = mesRef;
        comando.Parameters.Add(parameterMesRef);

        SqlParameter parameterAnoRef = new SqlParameter("@nano_ref", SqlDbType.NVarChar);
        parameterAnoRef.Value = anoRef;
        comando.Parameters.Add(parameterAnoRef);

        SqlParameter parameterValorFact = new SqlParameter("@nvalor", SqlDbType.Real);
        parameterValorFact.Value = Convert.ToDecimal(valorFact);
        comando.Parameters.Add(parameterValorFact);

        DateTime data = Convert.ToDateTime(venc);

        SqlParameter parameterVencimento = new SqlParameter("@nvencimento",
SqlDbType.DateTime);
        parameterVencimento.Value = data;
        comando.Parameters.Add(parameterVencimento);

        SqlParameter parameterObs = new SqlParameter("@ninfo", SqlDbType.NVarChar);
        parameterObs.Value = obs;
        comando.Parameters.Add(parameterObs);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int cadastraAreaPrivada(string nAreaPrivada, string nPredio, string nTempo,
string nValor)
    {
        SqlConnection ligacao = getConexao();
        SqlCommand comando = new SqlCommand("insert into
Detalhe_AreaPrivada(id_itemAreaPrivada,id_predio,tempoReserva, valorReserva) values
(@nAreaPrivada, @nPredio, @nTempo, @nValor)", ligacao);

        SqlParameter parameterAreaPrivada = new SqlParameter("@nAreaPrivada",
SqlDbType.Int);
        parameterAreaPrivada.Value = Convert.ToInt32(nAreaPrivada);
        comando.Parameters.Add(parameterAreaPrivada);

        SqlParameter parameterPredio = new SqlParameter("@nPredio", SqlDbType.Int);
        parameterPredio.Value = Convert.ToInt32(nPredio);
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterTempo = new SqlParameter("@nTempo", SqlDbType.NVarChar);

```

```

        parameterTempo.Value = nTempo;
        comando.Parameters.Add(parameterTempo);

        SqlParameter parameterValor = new SqlParameter("@nValor", SqlDbType.Real);
        parameterValor.Value = Convert.ToDecimal(nValor);
        comando.Parameters.Add(parameterValor);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int cadastraCotaCondo(string nidPredio, string nmesRef, string nanoRef, string
nValor, string nVencimento)
    {
        SqlConnection ligacao = getConexao();
        SqlCommand comando = new SqlCommand("insert into
CotaCondo(id_predio,mes_ref,ano_ref,valor,vencimento) values (@nidPredio, @nmesRef, @nanoRef,
@nValor, @nVencimento)", ligacao);

        SqlParameter parameterIdPredio = new SqlParameter("@nidPredio", SqlDbType.Int);
        parameterIdPredio.Value = Convert.ToInt32(nidPredio);
        comando.Parameters.Add(parameterIdPredio);

        SqlParameter parameterMesRef = new SqlParameter("@nmesRef", SqlDbType.NVarChar);
        parameterMesRef.Value = nmesRef;
        comando.Parameters.Add(parameterMesRef);

        SqlParameter parameterAnoRef = new SqlParameter("@nanoRef", SqlDbType.NVarChar);
        parameterAnoRef.Value = nanoRef;
        comando.Parameters.Add(parameterAnoRef);

        SqlParameter parameterValor = new SqlParameter("@nValor", SqlDbType.Real);
        parameterValor.Value = Convert.ToDecimal(nValor);
        comando.Parameters.Add(parameterValor);

        DateTime data = Convert.ToDateTime(nVencimento);

        SqlParameter parameterVenc = new SqlParameter("@nVencimento", SqlDbType.DateTime);
        parameterVenc.Value = data;
        comando.Parameters.Add(parameterVenc);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int moradorPagaCota(string valor, string nidPredio)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Predio] SET [fundo] = @fundo WHERE [id_predio] =
@nidPredio";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterIdPredio = new SqlParameter("@nidPredio", SqlDbType.Int);
        parameterIdPredio.Value = Convert.ToInt32(nidPredio);
        comando.Parameters.Add(parameterIdPredio);

        SqlParameter parameterFundo = new SqlParameter("@fundo", SqlDbType.Real);
        parameterFundo.Value = Convert.ToDecimal(valor);
        comando.Parameters.Add(parameterFundo);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int cadastraPgtoCota(string nUtilizador, string nCotaCondo, string nPredio,
string nEstado)
    {
        SqlConnection ligacao = getConexao();
        SqlCommand comando = new SqlCommand("insert into
PgtoCota(id_utilizador,id_cotaCondo,id_predio,statusPgto) values (@nUtilizador, @nCotaCondo,
@nPredio, @nEstado)", ligacao);

```

```

        SqlParameter parameterUtilizador = new SqlParameter("@nUtilizador", SqlDbType.Int);
        parameterUtilizador.Value = Convert.ToInt32(nUtilizador);
        comando.Parameters.Add(parameterUtilizador);

        SqlParameter parameterCotaCondo = new SqlParameter("@nCotaCondo", SqlDbType.Int);
        parameterCotaCondo.Value = Convert.ToInt32(nCotaCondo);
        comando.Parameters.Add(parameterCotaCondo);

        SqlParameter parameterPredio = new SqlParameter("@nPredio", SqlDbType.Int);
        parameterPredio.Value = Convert.ToInt32(nPredio);
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterEstado = new SqlParameter("@nEstado", SqlDbType.NVarChar);
        parameterEstado.Value = nEstado;
        comando.Parameters.Add(parameterEstado);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int atualizaStatusPgtoCota(string nUtilizador, string nCotaCondo, string
nPredio, string nEstado)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [PgtoCota] SET [statusPgto] = @statusPgto WHERE
[id_utilizador] = @id_utilizador AND [id_cotaCondo] = @id_cotaCondo AND [id_predio] =
@id_predio";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterUtilizador = new SqlParameter("@id_utilizador",
SqlDbType.Int);
        parameterUtilizador.Value = Convert.ToInt32(nUtilizador);
        comando.Parameters.Add(parameterUtilizador);

        SqlParameter parameterCotaCondo = new SqlParameter("@id_cotaCondo", SqlDbType.Int);
        parameterCotaCondo.Value = Convert.ToInt32(nCotaCondo);
        comando.Parameters.Add(parameterCotaCondo);

        SqlParameter parameterPredio = new SqlParameter("@id_predio", SqlDbType.Int);
        parameterPredio.Value = Convert.ToInt32(nPredio);
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterEstado = new SqlParameter("@statusPgto", SqlDbType.NVarChar);
        parameterEstado.Value = nEstado;
        comando.Parameters.Add(parameterEstado);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int atualizaFundoPredio(string endereco, string valorFundo)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Predio] SET [fundo] = @fundo WHERE [id_predio] =
@idPredio";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterEndereco = new SqlParameter("@idPredio", SqlDbType.Int);
        parameterEndereco.Value = Convert.ToInt32(endereco);
        comando.Parameters.Add(parameterEndereco);

        SqlParameter parameterValorFundo = new SqlParameter("@fundo", SqlDbType.Real);
        parameterValorFundo.Value = Convert.ToDecimal(valorFundo);
        comando.Parameters.Add(parameterValorFundo);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }
}

```

```

        public int atualizaStatusContaPredio(string idConta, string idPredio, string status,
string dataPgto)
        {
            SqlConnection ligacao = getConexao();
            string insere_sql = "UPDATE [Conta] SET [id_statusConta] = @statusConta, [dataPgto]
= @dataPgto WHERE [id_predio] = @idPredio AND [id_conta] = @idConta";
            SqlCommand comando = new SqlCommand(insere_sql, ligacao);

            SqlParameter parameterIdConta = new SqlParameter("@idConta", SqlDbType.Int);
            parameterIdConta.Value = Convert.ToInt32(idConta);
            comando.Parameters.Add(parameterIdConta);

            SqlParameter parameterEndereco = new SqlParameter("@idPredio", SqlDbType.Int);
            parameterEndereco.Value = Convert.ToInt32(idPredio);
            comando.Parameters.Add(parameterEndereco);

            SqlParameter parameterStatusConta = new SqlParameter("@statusConta",
SqlDbType.Int);
            parameterStatusConta.Value = Convert.ToInt32(status);
            comando.Parameters.Add(parameterStatusConta);

            DateTime data = Convert.ToDateTime(dataPgto);

            SqlParameter parameterDataPgto = new SqlParameter("@dataPgto", SqlDbType.DateTime);
            parameterDataPgto.Value = data;
            comando.Parameters.Add(parameterDataPgto);

            int num = comando.ExecuteNonQuery();
            closeConexao(ligacao);
            return num;
        }

        public int cadastraReserva(string utilizador, string area, string status, string dia,
string diaHoje)
        {
            SqlConnection ligacao = getConexao();
            SqlCommand comando = new SqlCommand("insert into
Reserva(id utilizador,id detalheAreaPrivada,status,diaReserva,diaHoje) values (@id_utilizador,
@id_detalheAreaPrivada, @status, @dia, @diaHoje)", ligacao);

            SqlParameter parameterUtilizador = new SqlParameter("@id_utilizador",
SqlDbType.Int);
            parameterUtilizador.Value = Convert.ToInt32(utilizador);
            comando.Parameters.Add(parameterUtilizador);

            SqlParameter parameterArea = new SqlParameter("@id_detalheAreaPrivada",
SqlDbType.Int);
            parameterArea.Value = Convert.ToInt32(area);
            comando.Parameters.Add(parameterArea);

            SqlParameter parameterStatus = new SqlParameter("@status", SqlDbType.NVarChar);
            parameterStatus.Value = status;
            comando.Parameters.Add(parameterStatus);

            DateTime data = Convert.ToDateTime(dia);
            DateTime hoje = Convert.ToDateTime(diaHoje);

            SqlParameter parameterDia = new SqlParameter("@dia", SqlDbType.DateTime);
            parameterDia.Value = data;
            comando.Parameters.Add(parameterDia);

            SqlParameter parameterDiaHoje = new SqlParameter("@diaHoje", SqlDbType.DateTime);
            parameterDiaHoje.Value = hoje;
            comando.Parameters.Add(parameterDiaHoje);

            int num = comando.ExecuteNonQuery();
            closeConexao(ligacao);
            return num;
        }

        public int atualizaEstadoReserva(string idReserva, string estado)
        {

```

```

        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Reserva] SET [status] = @status WHERE [id_reserva] =
@idReserva";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterIdReserva = new SqlParameter("@idReserva", SqlDbType.Int);
        parameterIdReserva.Value = Convert.ToInt32(idReserva);
        comando.Parameters.Add(parameterIdReserva);

        SqlParameter parameterStatus = new SqlParameter("@status", SqlDbType.NVarChar);
        parameterStatus.Value = estado;
        comando.Parameters.Add(parameterStatus);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int updateNovaSenha(string Utilizador, string Password)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Utilizador] SET [password] = @password WHERE
[id_utilizador] = @utilizador";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterUtilizador = new SqlParameter("@utilizador", SqlDbType.Int);
        parameterUtilizador.Value = Convert.ToInt32(Utilizador);
        comando.Parameters.Add(parameterUtilizador);

        SqlParameter parameterPass = new SqlParameter("@password", SqlDbType.NVarChar);
        parameterPass.Value = Password;
        comando.Parameters.Add(parameterPass);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int cancelaMoradorPosCondo(string idPredio, string nCadastro, string nInfo,
DateTime nDataFim)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Utilizador] SET [cadastro] = @cadastro, [dataFim] =
@dataFim , [info] = @info WHERE [id_predio] = @idPredio";
        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterPredio = new SqlParameter("@idPredio", SqlDbType.Int);
        parameterPredio.Value = Convert.ToInt32(idPredio);
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterCadastro = new SqlParameter("@cadastro", SqlDbType.NVarChar);
        parameterCadastro.Value = nCadastro;
        comando.Parameters.Add(parameterCadastro);

        SqlParameter parameterInfo = new SqlParameter("@info", SqlDbType.NVarChar);
        parameterInfo.Value = nInfo;
        comando.Parameters.Add(parameterInfo);

        SqlParameter parameterDataFim = new SqlParameter("@dataFim", SqlDbType.DateTime);
        parameterDataFim.Value = nDataFim;
        comando.Parameters.Add(parameterDataFim);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int atualizaPredioPossuiAreaPrivada(string idPredio, string status)
    {
        SqlConnection ligacao = getConexao();
        string insere_sql = "UPDATE [Predio] SET [possuiAreaPrivada] = @status WHERE
[id_predio] = @idPredio";

```

```

        SqlCommand comando = new SqlCommand(insere_sql, ligacao);

        SqlParameter parameterPredio = new SqlParameter("@idPredio", SqlDbType.Int);
        parameterPredio.Value = Convert.ToInt32(idPredio);
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterStatus = new SqlParameter("@status", SqlDbType.NVarChar);
        parameterStatus.Value = status;
        comando.Parameters.Add(parameterStatus);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }

    public int cadastraPgtoCotaUtilizador(string Cota, string Predio, string Utilizador,
    string Estado)
    {
        SqlConnection ligacao = getConexao();
        SqlCommand comando = new SqlCommand("insert into
PgtoCota(id_utilizador,id_cotaCondo,id_predio,statusPgto) values (@utilizador, @cota, @predio,
@estado)", ligacao);

        SqlParameter parameterUtilizador = new SqlParameter("@utilizador", SqlDbType.Int);
        parameterUtilizador.Value = Convert.ToInt32(Utilizador);
        comando.Parameters.Add(parameterUtilizador);

        SqlParameter parameterCota = new SqlParameter("@cota", SqlDbType.Int);
        parameterCota.Value = Convert.ToInt32(Cota);
        comando.Parameters.Add(parameterCota);

        SqlParameter parameterPredio = new SqlParameter("@predio", SqlDbType.Int);
        parameterPredio.Value = Convert.ToInt32(Predio);
        comando.Parameters.Add(parameterPredio);

        SqlParameter parameterStatus = new SqlParameter("@estado", SqlDbType.NVarChar);
        parameterStatus.Value = Estado;
        comando.Parameters.Add(parameterStatus);

        int num = comando.ExecuteNonQuery();
        closeConexao(ligacao);
        return num;
    }
}
}

```