

## **Desenvolvimento de sistema de gerenciamento de eventos agropecuários com uso de software open source**

**Augusto Zonta**

Agência Paulista de Tecnologia dos Agronegócios - Secretaria de Agricultura e Abastecimento do Estado de São Paulo.

**Maria Cristina de Mello Zonta**

Agência Paulista de Tecnologia dos Agronegócios - Secretaria de Agricultura e Abastecimento do Estado de São Paulo.

### **Resumo**

A Agência Paulista de Tecnologia dos Agronegócios é uma das coordenadorias da Secretaria de Agricultura do Estado de São Paulo e conta com seis institutos de pesquisa e quatorze Polos regionais distribuídos estrategicamente pelo estado. Tem a missão de coordenar, gerar e transferir conhecimento científico e tecnológico voltado para o agronegócio, promovendo o desenvolvimento sócio econômico e o equilíbrio do meio ambiente. Anualmente a APTA realiza cerca de 90 eventos que atendem a mais de 7000 produtores, profissionais, técnicos, universitários e a comunidade em geral. O objetivo deste trabalho foi criar um sistema de gerenciamento de eventos utilizando-se de softwares *open source* com licenças permissivas que reduza drasticamente o tempo gasto para as inscrições, que seja de fácil operação, ajude na análise do perfil dos participantes, na avaliação da qualidade de divulgação do evento e que ofereça emissão e correção dos certificados de forma rápida e descomplicada. O sistema gerado foi desenvolvido em linguagem Python 2.7.x e permitiu a melhoria dos serviços de capacitação e transferência de conhecimento realizado pela instituição reduzindo o tempo gasto para cada 100 inscrições em 70 minutos. Conclui-se que é possível elaborar um produto final de qualidade com o uso de software de código aberto sem nenhuma modificação nos seus respectivos códigos fontes, simplesmente os utilizando como estão.

Palavra-chave: engenharia de software, open source, python, apta

### **Introdução**

A Agência Paulista de Tecnologia dos Agronegócios é uma das coordenadorias da Secretaria de Agricultura do Estado de São Paulo e conta com seis institutos de pesquisa e quatorze Polos regionais distribuídos estrategicamente pelo estado. Tem a missão de coordenar, gerar e transferir conhecimento científico e tecnológico voltado para o agronegócio, promovendo o desenvolvimento sócio econômico e o equilíbrio do meio ambiente (APTA, 2015). O Núcleo de Informação e Transferência do Conhecimento é responsável por planejar, organizar e executar estas ações de formação de recursos humanos.

Anualmente a APTA realiza cerca de 90 eventos atendendo mais de 7000 produtores, profissionais, técnicos, universitários e a comunidade em geral. Devido a grande demanda por informações técnicas sobre as diversas cadeias produtivas do agronegócio, a procura pelos eventos realizados é grande e muitas vezes processo de inscrição é lento, causa desgastes e falhas no preenchimento das fichas. As falhas mais comuns são fichas incompletas, ilegíveis ou com erros de compreensão dos campos.

Sistemas informatizados de gerenciamento são particularmente úteis quando objetiva-se reduzir o tempo de execução de um processo e mitigar erros causados durante a execução de tarefas repetitivas.

## Revisão bibliográfica

Basicamente código fonte é um texto com uma sintaxe específica contendo instruções que definem como um software deve se comportar. Antes dos anos oitenta os usuários tinham apenas acesso a softwares proprietários mediante ao pagamento de taxas de licença para uso, atualizações e manutenção. Estes softwares não permitiam qualquer tipo de alteração e só podiam ser utilizados em condições restritas impostas por suas licenças.

O conceito de software livre surgiu em 1984 com uma forte carga política. A princípio associava-se software livre com a ideia de gratuidade, mas segundo o fundador deste movimento, Richard Stallman, o conceito se refere à liberdade (FSF, 2015). Liberdade de executar o software para qualquer propósito, liberdade para estudar o código fonte, liberdade para adaptar o software as suas necessidades e liberdade para distribuir este software. Porém o software derivado de um software livre é obrigado a transmitir estas mesmas condições de liberdade. O software livre passou a ser visto por alguns como uma ameaça ao software proprietário e seus oponentes questionavam sua segurança e qualidade.

Com o passar dos anos alguns desenvolvedores que apoiavam o software livre questionaram o chamado “efeito viral” da licença, que impedia a liberdade de se escolher uma nova forma de licenciamento para os softwares criados. Em 1997, com o intuito de amenizar estas questões ideológicas e o conflito com as empresas de software proprietário, um novo grupo se formou propondo o termo software *open source* (OSI, 2015). Focado na visão técnica de que um software com código aberto é mais eficiente devido ao trabalho colaborativo e multidisciplinar dos seus desenvolvedores.

No entanto, para alguns, ainda hoje permanece a dúvida se é possível desenvolver um software sustentável, de qualidade e seguro utilizando-se de softwares com código aberto (Saleh. A, 2004).

## Objetivos

Para o desenvolvimento deste sistema o Núcleo de Informação e Transferência do Conhecimento da APTA definiu alguns critérios que deveriam ser atendidos:

### **Objetivos gerais**

- Desenvolver um sistema para controle de eventos com software *open source*.
- Ser isento de taxas de manutenções, autorizações ou de atualizações periódicas.
- Ser flexível, passível de modificações ou implementação de novos recursos.
- Melhorar significativamente o tempo de inscrição de um participante em comparação ao método tradicional com fichas de papel.
- Seus recursos devem auxiliar as análises realizadas pelo Núcleo de Informação da APTA.

### **Objetivos específicos**

- Auxiliar na tomadas de decisões gerenciais com o acompanhamento da disponibilidade de vagas.
- Apresentar em tempo real o perfil dos participantes e as cidades atendidas.
- Realizar todas suas operações sem a necessidade de conexão com a internet ou intranet. Sendo desta forma particularmente útil para eventos agropecuários longe do alcance do sinal de wi-fi ou cabeamento.

### **Materiais e métodos**

Para garantir qualidade no desenvolvimento do sistema seguimos um modelo de engenharia de software em cascata (Pressman. R, 2011) composto pelas seguintes etapas: Análise do problema, requisitos a serem atendidos, planejamento, codificação, teste e operação. Segundo o autor a execução cuidadosa de cada etapa assegura o sucesso da etapa seguinte.

- Análise: Observar o problema de forma ampla, detectando as necessidades que precisam ser atendidas e também suas interações com os elementos humanos externos.
- Requisitos: As metas principais para a resolução do problema. Definir as funcionalidades do sistema.

- Planejamento: Esquemas, croquis e fluxograma do futuro sistema. Definir a linguagem e as ferramentas mais adequadas e estratégias para futuras ampliações de funcionalidades.
- Codificação: É a programação propriamente dita na linguagem e com as ferramentas de escolha.
- Teste: Nesta etapa o desenvolvedor observa os resultados, o desempenho e se há necessidade de ajustes na interface ou na mecânica dos processos.
- Operação: Execução do sistema em uma situação real de trabalho.

### **Softwares utilizados no desenvolvimento**

#### Python 2.7.x – Licença: PSF

Python é uma linguagem de programação desenvolvida em 1991, por Guido Van Rossum. De alto nível e baixa curva de aprendizado, possui uma sintaxe limpa, clara e de fácil leitura, produzindo-se mais e escrevendo menos em pouco tempo de estudo quando comparada a outras linguagens mais tradicionais. Estas características tornaram o Python muito popular em poucos anos. Centenas de módulos, ou bibliotecas, foram criadas para ampliar suas capacidades. Outro fator importante para os que se iniciam nesta linguagem é a grande quantidade de documentação disponível na internet e uma comunidade ativa. A sua principal desvantagem é a baixa velocidade quando comparada a linguagens compiladas como C++.

#### wxPython 2.8 – Licença: wxWindows

wxPython é uma biblioteca de classes amplamente utilizada no desenvolvimento de interfaces gráficas do usuário e foi criada utilizando o wxWindows em C++ como base. Realiza chamadas a funções definidas pelo programador como resposta a eventos assíncronos realizados pelo usuário ou pelo próprio sistema. Oferece grande liberdade ao desenvolvedor quando é utilizada sem modificações e o software criado distribuído em formato binário.

#### SQLite 3.0 – Licença: Domínio Público

SQLite é um banco de dados muito compacto e que não necessita de conexão ao servidor. Ele lê e escreve os dados diretamente em um único arquivo local. É plataforma independente e amplamente utilizado também em sistemas embarcados. Permite importar e exportar os dados facilmente em formato Comma-separated Values (CSV).

### Matplotlib 1.2 – Licença: Python Software Foundation

Matplotlib cria gráficos e histogramas com qualidade e total controle. Possui sintaxe semelhante ao MATLAB. Com poucas linhas de código produzem-se claras visualizações dos dados que podem ser salvos facilmente em formato PS, SVG ou PNG.

### Reportlab 2.7 – Licença: BSD

Reportlab permite a criação de documentos em PDF com qualquer nível de complexidade gráfica. É possível inclusive utilizar *templates* para a criação automatizada de diversos relatórios.

### Python Image Library 1.1 – Licença: BSD

A biblioteca PIL adiciona ao Python a capacidade de importar, exportar, manipular e processar imagens em mais diversos formatos de arquivo.

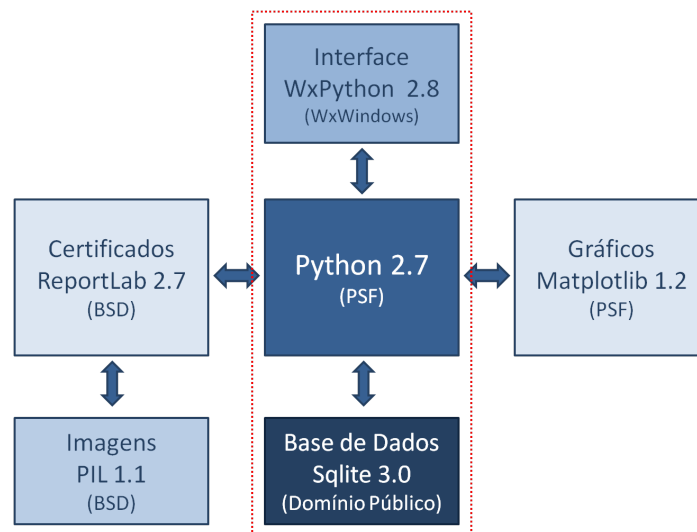


Figura 1: Esquema do sistema e suas inter-relações.

### O fluxo na execução de um evento

As etapas e decisões durante a execução de um evento:

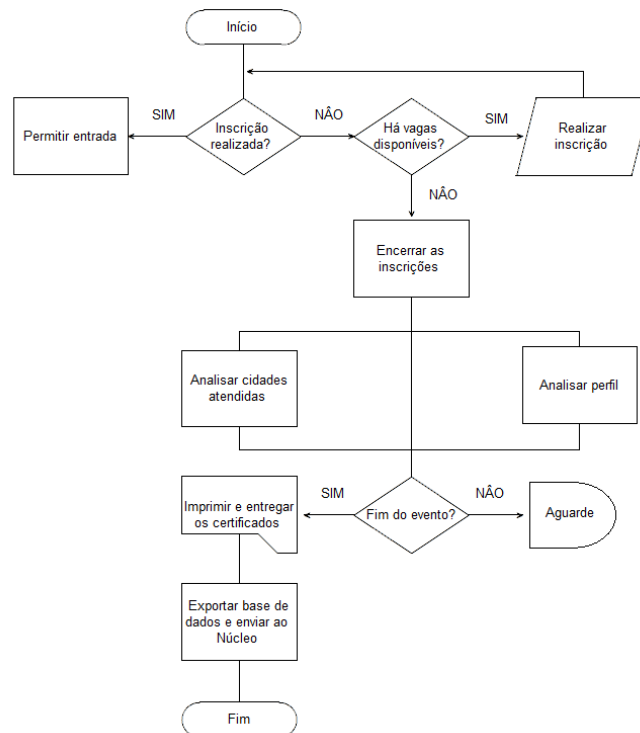


Figura 2: Fluxograma da execução de um evento.

## Resultados

### Design e modularidade

Uma interface intuitiva é resultado de uma representação gráfica simplificada de processos complexos combinados, baseado na percepção humana. Esta linguagem visual é que permite o eficaz fluxo de informações entre o meio humano e a máquina.

O sistema foi planejado utilizando-se a classe `wx.Notebook` do `wxPython` com as abas posicionadas horizontalmente pensando no fluxo da informação (Figura 1). A classe `wx.Notebook` gerencia múltiplas janelas que são posteriormente associadas as abas. Este característica possibilitou que o código fosse escrito de forma modularizada, onde cada aba pode ser executada, fora do notebook, individualmente pelo desenvolvedor durante a depuração de erros. E novas abas podem ser incorporadas facilmente, adicionando recursos extras ao sistema, sem a necessidade de alterações na estrutura central.

```
class MainPanel(wx.Panel):  
    def __init__(self, parent):  
        wx.Panel.__init__(self, parent)
```

```
self.nb = wx.Notebook(self)

self.pages = [
    (Cadastro.MainPanel(self.nb), "Cadastro"),
    ...
    (Certificados.MainPanel(self.nb), "Certificados"),
]

for self.page, self.label in self.pages:
    self.nb.AddPage(self.page, self.label)
```

### Cadastro dos participantes

As fichas de papel tradicionalmente utilizadas para as inscrições eram preenchidas pelos próprios participantes e entregues à atendente que as examinava uma a uma. Os certificados seriam em seguida confeccionados com o uso do processador de texto. Além da lentidão a maior dificuldade deste método é entender o escrito devido ao baixo nível de escolaridade de alguns participantes. Isto ocasionava erros frequentes na emissão dos certificados. Em eventos com mais de cem participantes ficava claro a necessidade do uso de um sistema informatizado.

A aba de cadastro contém os mesmos campos das fichas de papel. Os campos podem ser navegados com o uso do mouse ou sequencialmente com a tecla tab. Para os campos “perfil do participante” e “modo de participação”, contendo cinco opções em cada um, o elemento de interface escolhido foi o menu do tipo *dropdown*, uma vez que economiza espaço na interface e não sobrecarrega visualmente o usuário (Figura 3). Após avaliar o uso do sistema por um período de um ano, sendo utilizado em diferentes eventos por diferentes usuários, observou-se que o tempo médio de inscrição de um participante foi em torno de 30 segundos. Isto significou uma redução do tempo de 70 minutos para cada 100 inscritos quando comparado a tradicionais fichas de papel.

Figura 3: Interface de cadastro dos participantes.

### Banco de dados

Uma vez que a nossa aplicação não é complexa e que um dos objetivos é que o sistema possa ser utilizado em locais sem acesso à internet ou intranet, não houve a necessidade de um poderoso banco de dados cliente/servidor para o armazenamento das informações. O banco de dados Sqlite foi a escolha mais adequada para este caso, uma vez que ele capaz de criar, ler e escrever diretamente para o arquivo de banco de dados. Possui recursos de linguagem de consulta estruturada (SQL) reduzidos, porém sua simplicidade e tamanho são seus pontos fortes. Não necessita de instalação, configuração ou manutenção e não possui dependências externas.

Suas operações seguem o conceito de Atomicidade, Consistência, Isolamento e Durabilidade. Desta forma todas as transações da chamada devem apresentar sucesso ou serão abortadas. O banco de dados não recebe operações parciais e as chaves primárias devem ser consistentes e únicas. Operações intermediárias que ocorrem paralelamente em sistemas de multi usuários estão isoladas até o sucesso da operação. Por fim as informações inseridas são persistentes e duráveis.

Inicialmente cria-se o arquivo do banco de dados (Figura 4) e as funções para as quatro operações básicas: Adicionar, Ler, Alterar e Remover. O método Bind() do wxPython nos permite associar estas funções aos respectivos botões e ao manipulador de eventos.

```
conn = sqlite3.connect("Dados.db")  
curs = conn.cursor()
```



```
self.newbtn = wx.Button(self, -1, "Novo", size=(66, 28))
self.Bind(wx.EVT_BUTTON, self.newrec, self.newbtn)

def newrec(self,event):
    self.nome = self.campo1.GetValue()
    ...
    self.cnpj = self.campo7.GetValue()

    self.holder = (self.nome, self.email, self.fone, self.cidade, self.inst,
self.perfev, self.cnpj, self.tipoev, self.chboxvar,)

    self.sql = "insert into cadastro (nome, email, fone, cidade, inst, perfil,
cnpj, tipo, cert) values (?, ?, ?, ?, ?, ?, ?, ?, ?)"

    curs.execute(self.sql,self.holder)

    conn.commit()

    return


conn.close()
```

O uso de strings concatenadas em operações SQL não é recomendado, pois adiciona uma vulnerabilidade que permite a um usuário mal intencionado injetar ataques ao banco de dados obtendo acesso a informações restritas ou apagar todos os registros. Portanto, pensando em futuramente adicionar ao sistema recursos para o trabalho em rede, decidimos manter a segurança e fazer uso de *placeholders* para receber os argumentos em todas as transações. Os locais com o símbolo “?” estão reservados para receber apenas os valores passados pelas variáveis.

### **Acompanhamento das inscrições**

A aba “listagem” apresenta o número de participantes cadastrados, ordenados alfabeticamente, independente da ordem de inscrição e as linhas possuem cores alternadas para facilitar a localização. O email e telefone estão

seguidos ao nome. Este arranjo é útil nos casos de inscrições antecipadas e que se precisa entrar em contato com os inscritos para informar alguma alteração do evento (Figura 5).



N.	Nome	Email	Fone	Perfil
1	Dom Pedro I	dp@net.br	(11)98765-3241	Extensão
2	Mestre Zulu	msz@net.br	(11)98765-2143	Produtor
3	Pedro Alvares Cabral	cabral@net.br	(11)98765-4321	Pesquisa

Figura 5: Aba para o acompanhamento do número de inscritos.

A classe `wx.ListCtrl` cria listas de forma muito direta, basta definir uma nova lista e adicionar as colunas. Uma chamada SQL preenche a lista criada.

```
self.list_ctrl = wx.ListCtrl(self, size=(-1,-1), style=wx.LC_REPORT |  
wx.BORDER_SUNKEN)  
  
self.list_ctrl.InsertColumn(0, 'N.', width=35)  
self.list_ctrl.InsertColumn(1, 'Nome', width=145)  
self.list_ctrl.InsertColumn(2, 'Email', width=150)  
self.list_ctrl.InsertColumn(3, 'Fone', width=115)  
self.list_ctrl.InsertColumn(4, 'Perfil', width=70)  
  
self.titulo = wx.StaticText(self, -1, "Lista de Participantes",  
style=wx.ALIGN_CENTRE)  
  
...  
  
self.sql = cursor.execute("SELECT * FROM cadastro ORDER BY nome collate  
nocase")
```

## Gráficos de cidades e perfil

Durante as inscrições o preenchimento de certos campos é obrigatório. Com o campo “cidade” é possível avaliar a eficiência dos meios de divulgação utilizados bem como a dispersão do conhecimento oferecido no evento. O campo “perfil” permite avaliar se o evento atendeu o público alvo ou qual tipo de público se interessou pelas informações oferecidas. Ambos são de grande importância para a tomada de decisões gerenciais nos eventos futuros (Figura 6).

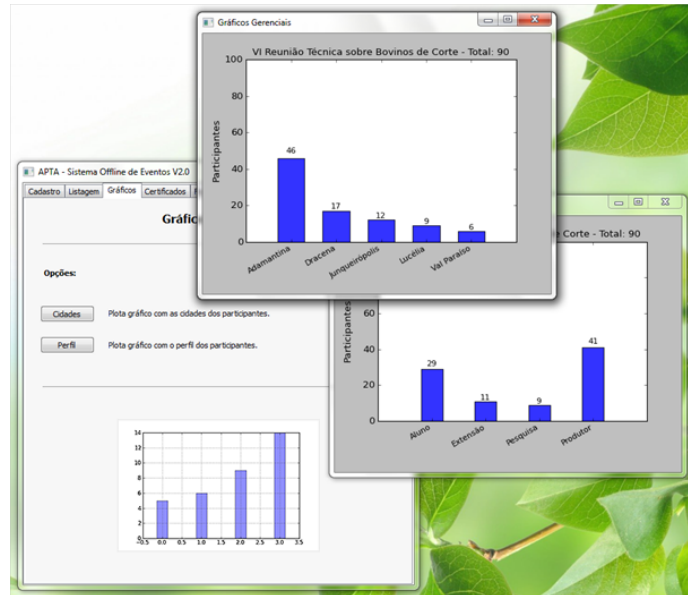


Figura 6: Gráficos gerenciais com as cidades atendidas e perfil de participantes.

Os dados de interesse são selecionados por uma chamada SQL e enviados para um loop FOR onde as variáveis xvalor e yvalor são utilizadas na plotagem com o método bar(). Os gráficos são exibidos em janelas independentes e são salvo automaticamente na área de trabalho em formato PDF para futuras consultas.

```
self.fig = Figure()
self.axes = self.fig.add_subplot(111)
self.canvas = FigCanvas(self, -1, self.fig)
self.axes.set_ylabel("Participantes")
self.axes.set_ylim(0,100)

self.bars = self.axes.bar(left=self.xvalor, height=self.yvalor, width=0.6)
self.axes.set_xticklabels(self.cidadelbl, fontsize=10)
```

```
self.canvas.draw()
```

### Emissão dos certificados

Os certificados são criados com a biblioteca Reportlab utilizando-se de um template que é preenchido por uma chamada SQL. Pode-se definir uma imagem de fundo para o uso de papel não previamente timbrado pela gráfica. O arquivo gerado é um PDF que pode ser impresso em qualquer impressora sem perda de formatação e os certificados entregues ao final do evento. Ou o arquivo pode ser enviado posteriormente para o email dos participantes. O Reportlab é muito rápido e o hardware não influencia tanto em seu desempenho. Em situação real de trabalho consegue criar 300 certificados em cerca de quinze segundos rodando com um hardware mediano. A biblioteca PIL foi utilizada para ajustar a nitidez, o brilho, o contraste, dimensionar e recortar as assinaturas, utilizadas nos certificados, que são fotografadas e inseridas no sistema.

```
conn = sqlite3.connect('Dados.db')
curs = conn.cursor()

self.certs = "SELECT * FROM cadastro where cert=1 order by nome
collate nocase"

curs.execute(self.certs)

self.total = curs.fetchall()
self.linhas = len(self.total)

...

self.canv =
canvas.Canvas(os.environ["USERPROFILE"]+"\Desktop\Certificados.pdf",
pagesize=landscape(A4))

self.contador = 0

while self.contador < self.linhas:
```

```
self.corpo = "Certifico que "+self.total[self.contador][1]+"  
participou, como "+self.total[self.contador][8]+" , "+self.nome+" , promovido pela  
APTA - "+self.local+" , com carga horária de "+self.carga+"."  
  
self.formatar = Paragraph(self.corpo, self.estilo)  
  
self.template = [ [self.espaco], [self.corpo], [self.espaco],  
[self.data], [self.assina], [self.codigo] ]  
  
self.dist = [7*cm, 4*cm, 1*cm, 1*cm, 4*cm, 2*cm]  
  
self.doc = Table(self.template, 24*cm, self.dist, splitByRow=True,  
repeatRows=1)  
  
self.doc.wrapOn(self.canv, 19*cm, 29*cm)  
  
self.doc.drawOn(self.canv, 4.5*cm, 1*cm)  
  
self.canv.showPage()  
  
self.contador = self.contador+1  
  
  
self.canv.save()
```

### **Ferramentas administrativas**

Ao final do evento os dados podem ser exportados em formato CSV com a finalidade arquivamento ou importação em outra base de dados. Também é possível imprimir a lista de presença para colher as assinaturas durante a entrega dos certificados (Figura 7).

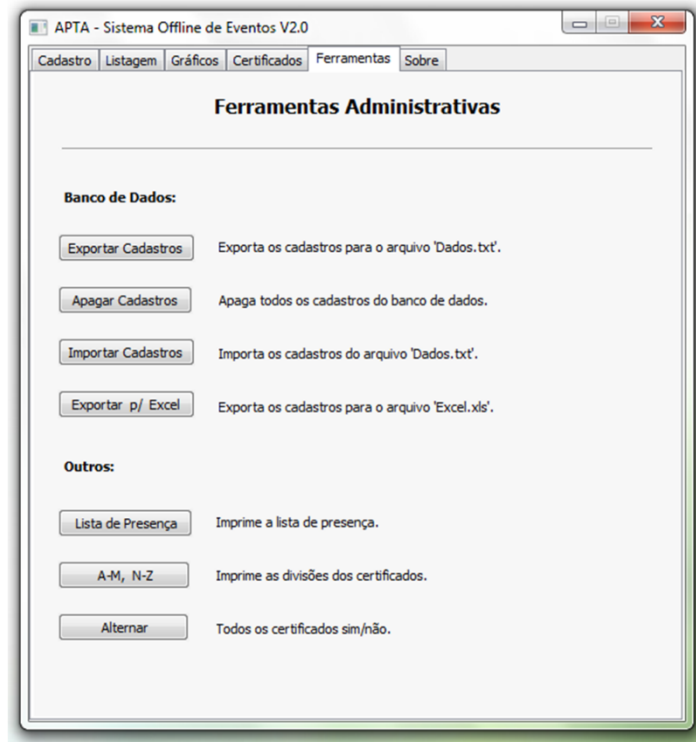


Figura 7: Ferramentas administrativas.

## Conclusão

O sistema de gerenciamento de eventos é um software estável, desenvolvido utilizando-se exclusivamente de software open source. Foi testado por um período de três anos sem apresentar nenhum problema técnico. E atendeu aos requisitos mínimos propostos pelo Núcleo de Informação e Transferência de Conhecimento da Agência Paulista de Tecnologia dos Agronegócios proporcionando uma melhoria na qualidade dos serviços prestados com a redução no tempo das inscrições aliada ao menor apresentação de erros na confecção dos certificados.

## Referências bibliográficas

Agência Paulista de Tecnologia dos Agronegócios. Disponível em: <<http://www.apta.sp.gov.br/>>. Acesso em: 01 dez. 2015

SALEH, A. M. Adoção de tecnologia: um estudo sobre o uso de software livre nas empresas. 2004. USP, São Paulo, 2004. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/12/12139/tde-06122004-123821/pt-br.php>>. Acesso em: 01 dez. 2015

Free Software Foundation. Disponível em: <<https://www.fsf.org/>>. Acesso em: 01 dez. 2015

Open Source Initiative. Disponível em: <<http://www.opensource.org/>>. Acesso em: 01 dez. 2015

PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011. 780p.

Python Software Foundation. Disponível em:<<https://www.python.org/>>. Acesso em: 01 dez. 2015